Autonomous Rovers for Mars Exploration[§]

Richard Washington[1], Keith Golden, John Bresina, David E. Smith, Corin Anderson[2], Trey Smith[3],
NASA Ames Research Center, MS 269-2
Moffett Field, CA 94035
650-604-5000
{richw | kgolden | bresina | de2smith | corin | trey}@ptolemy.arc.nasa.gov

ABSTRACT—The Pathfinder mission demonstrated the potential for robotic Mars exploration but at the same time indicated the need for more robust rover autonomy. Future planned missions call for long traverses over unknown terrain, robust navigation and instrument placement, and reliable operations for extended periods of time. Ultimately, missions may visit multiple science sites in a single day and perform opportunistic science data collection, as well as complex scouting, construction, and maintenance tasks in preparation for an eventual human presence. Significant advances in robust autonomous operations are needed to enable these types of missions.

Towards this end, we have designed an on-board executive architecture that incorporates robust flexible operation, resource utilization, and failure recovery. In addition, we have designed ground tools to produce and refine contingent schedules that take advantage of the on-board architecture's flexible execution characteristics. Together, the on-board executive and the ground tools constitute an integrated rover autonomy architecture.

TABLE OF CONTENTS

1. INTRODUCTION

Mars exploration is a high priority at NASA. Current plans call for a mission to Mars every 26 months. Future missions will require rovers with more capabilities than have ever been demonstrated on previous missions. The rover to be launched in 2003 will be expected to survive for more than a year and traverse more than 10 kilometers. By contrast, the Sojourner rover in the 1997 Mars Pathfinder mission was only expected to survive for a few weeks and stayed within sight of the lander. Ultimately, in support of human exploration, rovers will be needed to scout out landing sites, find water and other resources, set up power plants, and mine and transport resources.

These capabilities will require rovers with robust autonomous operations so that they can perform independently over long intervals while achieving mission and science goals. The rovers will communicate infrequently with earth, and the communication will have high latency and low bandwidth. Rovers that are continually dependent on commands from Earth would incur huge cost and would achieve a much lower science return due to the lost time that the rover is waiting for instructions. Given that rovers have a limited lifetime, such wasted opportunity translates to much lower return on investment.

Robust autonomous operations will also be important in support of human presence on Mars. Even though latency will be less of a problem, the crew will be busy with more important tasks and will not have the time to continuously monitor the rovers and control their every move. Thus, autonomous rovers can greatly amplify the productivity of the small crew. It is also advantageous for the rovers to be responsible for their own well-being: to avoid getting damaged, as well as to diagnose and correct recoverable software and hardware failures. With robust autonomous rovers, the crew does not have to spend time baby-sitting rovers and can concentrate on survival and science.

Although much research has been done to endow autonomous robots with far greater capabilities than were demonstrated on Sojourner, little of that can be used directly for planetary missions. On space missions, safety is paramount. Missions are very expensive, and even the simplest mistakes can cause the mission to fail. If the rover became permanently stuck on a rock or damaged, the effect on the mission would be disastrous. Thus, on-board technology will always be more conservative than what has been tried in a laboratory setting.

In the following section, we discuss the capabilities of current rovers and the additional capabilities needed to support planned missions. In Section 3, we discuss a rover autonomy architecture that provides these needed capabilities. In Section 4, we discuss plans for an upcoming field test that will demonstrate some of these capabilities. We then discuss future directions of this work.

---

[1] Caelum Research Corporation at NASA Ames Research Center
[2] University of Washington
[3] Carnegie Mellon University

2. CURRENT AND FUTURE ROVER AUTONOMY

Criteria for robust autonomous rovers

For a rover to demonstrate robust autonomous operations, it will need to adapt to its environment. Although someday this may be performed through automatic, on-board modification of plans and models, this could produce unanticipated behaviors and thus would introduce an added risk into the mission. Instead, we focus on those aspects that can allow the rover controllers on ground to specify the rover's response to a range of possible operating situations that may arise. In particular, we consider the following criteria essential for robust rover autonomy:

- **Robust flexible operation**. A rover should choose its behavior based on the execution environment. The rover operators should be able to specify multiple execution behaviors along with the conditions under which they are appropriate.

- **Resource utilization**. Resources such as power, data storage, and communication bandwidth place restrictions on what the rover can do. In order to perform its science and mission objectives efficiently, the rover needs to make full use of the available resources, even when they change from expected values. To do this it should track actual and expected resources, signal resource conflicts, and allow its behavior to be specified with respect to the resources.

- **Failure recovery**. Because of the complexity of a rover's interactions with its environment, it will need the ability to recover from faults or anomalous situations. The rover should recover from those faults from which it safely can, and it should gather relevant information for problems that need to be handled by rover operators. Each of these will save command cycles and increase the efficiency of the rover.

In addition to these criteria, there are other considerations that affect planetary rover operations in general. These considerations influence the design of missions and consequently the work described in this paper:

- **Uncertainty**. Rovers operate in environments that are largely unknown. Techniques that require precise information about the environment to perform actions will be prone to failure.

- **Need for safety**. Space missions are expensive and rover failure irreparable. A planetary rover must work the first time, and it needs to guarantee its own safety in order to achieve the mission goals.

- **Limited communication**. With limited use of the Deep Space Network and with other communication limits imposed by orbiter schedulers, the rover must operate with relatively little and infrequent interaction with the rover operators. The rover must operate between communication events with no human intervention and no assistance from ground-based programs.

- **Need for understandability**. Mission designers must be convinced that the proposed approach will improve the quality of the mission. If the approach is complex and unclear to the mission designer, it is unlikely to be used.

- **Multiple objectives**. A deployed planetary rover must balance considerations such as navigation with science, communication, resource consumption, and fault recovery. A rover must be able to operate as an integrated system.

Mission-ready rovers

Examples of mission-ready rover technology are scarce. The primary source is the Mars rover development program at JPL, which produced the 1997 Sojourner rover. Sojourner is the only deployed planetary rover to demonstrate a significant level of autonomous operations [1]. Sojourner operated in a semi-autonomous mode, with periodic communication with Earth via the Pathfinder lander. Sojourner weighed about 23 pounds, operated for 83 sols (Martian days, which are 37 minutes longer than days on Earth), traveled 100 meters, performed 16 experiments on soil and rocks, and transmitted 550 images.

The Soviet space program produced two lunar rovers that were teleoperated from Earth in 1970 and 1973. The level of autonomy was basic, limited to stopping in response to excessive tilt, wheel blockage, or motor overheating. The teleoperation allowed the rovers to travel over 30 times further per day than Sojourner and perform nearly an order of magnitude more experiments per day, all this 20 years earlier.

Teleoperation is impractical at longer distances, so some level of autonomous operation is necessary for planetary rovers deployed beyond our moon. Current rover missions, such as Pathfinder, depend almost entirely on ground-based commanding and employ only enough on-board autonomy to safely follow uplinked commands. If anomalous situations arise, the rover goes into a safe mode and waits for an updated command sequence to be generated by the ground operations team and uplinked. In practice, this approach leads to high mission cost and missed science opportunities. The ground operations team for the Pathfinder mission had to adjust themselves to Mars time, sleeping and waking 37 minutes later each day, for the entire mission and had much more access to expertise in diagnosing and debugging problems than will be possible in future missions. With upcoming missions intended to last a year or more, expecting people to work such hours and maintain such vigilance is not feasible. Furthermore, since almost all of the intelligence behind the Sojourner rover was on Earth, the command sequences that Sojourner executed were quite fragile. There were many cases when something went wrong, and the team on Earth had to spend a day or more diagnosing and fixing the problem.

NASA and the European Space Agency (ESA) have published mission designs calling for autonomous Mars rovers [15,17]. The space agencies recognize the need for integrated architectures that allow for significant levels of autonomy in order to operate effective missions. In the 1996 field tests of Sojourner-style technology, three of the lessons learned dealt with a need for a higher degree of autonomous operations [2]:

- Science operations required higher than expected communication cycles. Need to improve rover autonomy for science operations.
- Operator interface need[s] to be improved to reduce operator load and to provide error free operations.
- Rover autonomous navigation not adequate for Viking Lander II type terrains. Improvements in autonomous navigation are required.

The second of these is not directly related to autonomous operations, but rather indicates the need for a combination of ground-based tools and on-board autonomy that will reduce the burden on the rover operators.

The ability of future rovers to achieve the more ambitious goals of future missions will depend critically on robust autonomous operation. Each of the criteria for robust autonomy can be seen to be lacking to some extent with current mission-ready rovers:

Robust flexible execution—Currently, a rover command sequence is specified to the lowest level of detail and leaves few, if any, choices to be made at execution time; hence, it admits only one, or a very small number of, valid execution behaviors. This simplifies the execution process but does not allow execution to be responsive to the dynamic status of the rover and the environment. This inflexibility can cause reduced productivity and execution failures.

For example, in Sol 22 of the mission, Sojourner received the following challenging instruction sequence:

1. Back up to the rock named Soufflé.
2. Place the arm with the spectrometer on the rock.
3. Do extensive measurements on the rock surface.
4. Perform a long traverse to another rock.

At the next communication opportunity, the news came in from Mars. The good news was that the sequence was executed to completion, including the longest traverse ever done in one day, a world's record for Mars. However, there was bad news with the good. The rover stopped short of the rock and the spectrometer was left hanging out in mid-air rather than placed on the rock. The rover sensors indicated that the spectrometer was not in contact with the rock, but the rover continued with its spectral measurements. Not only was the spectrometer data useless, but the rover spent over six hours running the spectrometer, using time and energy that could have been used otherwise. The rock was never again visited and thus that science opportunity was lost. A more desirable behavior would be for the rover either to stop and take pictures to aid the ground team's diagnosis of the placement problem or to give up on that science goal and continue with another.

Resource utilization—The generation of uplink sequences is based on estimated profiles, over time, of capacity and demand for each resource. However, there is great inherent uncertainty in the operating environment and its impact on performance of rover hardware components. For example, the power demand of a traversal is highly dependent on the incline, roughness, and traction of the terrain. Without an accurate estimate of power demand, battery state of charge at a given time cannot be accurately predicted. Furthermore, battery charge will also depend on how much time is spent in shade and on the tilt of the solar panels, which in turn depends on the surrounding landscape. In response to this uncertainty, worst-case estimates of resource usage and availability are typically used; however, even tight worst-case bounds may be difficult to predict. Because demand estimates are too pessimistic, execution of such sequences often results in reduced rover productivity through wasted time and lost opportunities. On the other hand, overly optimistic estimates of resource usage and availability result in higher risk to rover safety and a greater chance of broken plans.

Failure recovery—Current rovers have very limited capabilities for recovering from faults or anomalous situations. The rover's response to most execution failures is to halt all activity and wait for the ground operations team to determine what went wrong and uplink a recovery plan. Depending on the nature of the anomalous situation and the quality of the downlinked information for the purposes of diagnosis, this ground-based recovery process can cost days of rover idle time and lost science opportunity. For example, a rover traverse may fail with a high wheel current combined with the wheel encoder showing no movement. In this case, the wheel may be stuck on a rock or the encoder may be broken. The ground team will need to uplink diagnostic sequences to determine which one is the case, wait for the results to be downlinked, and eventually uplink recovery sequences to remove the rover from the rock if that is the problem. Valuable science opportunities are lost during this time. Even transient problems such as an overheated motor can cause plan failure, where perhaps a brief pause to cool down would be sufficient to remedy the problem.

In the 1996 Rocky 7 field tests, rocks got caught in the wheels, causing wheels to jam and drag on the ground [2,3]. This was diagnosed in the desert by human observers noticing the track that the stuck wheel left in the soil, and it was fixed by prying out the rock with tools. On a more remote site, an automatic fault diagnosis and recovery system would allow the rover to continue, or at least to stop before damaging itself and to request help from Earth.

Research on outdoor robots

A number of rovers have been built for Earth-based missions or to test research ideas on realistic platforms. Much of the work on outdoor robotics has focused on advances in the robotic hardware necessary to operate in the unstructured,

difficult environment that an outdoor setting presents. For example, the main contribution of the Ambler robot is a novel form of leg-based locomotion [4]. This work is complementary to the work in this paper, which concentrates on the software architecture needed for robust autonomous operations.

Many outdoor robots are designed to be largely teleoperated [5,6,7], concentrating on the ability to navigate autonomously to a given waypoint, avoiding obstacles along the way. The performance of these robots is impressive in terms of distance traveled [6] and environmental hazards overcome [7], but the problem was restricted so that the robots did not have to exhibit robust behavior with respect to science goal achievement.

The ADS autonomous land vehicle [8] generated contingent plans over uncertain terrain. The work was restricted to path planning, and a prior, possibly inaccurate map of the environment was assumed to exist for generating paths.

Research on indoor robots

In contrast to the deployed missions and even outdoor robots, results from research labs on indoor robots would suggest that mobile rovers can operate reliably and autonomously over a wide range of conditions and a wide range of time, demonstrating impressive abilities for navigation, fault recovery, replanning, and science acquisition. However, the state of the art in outdoor rover technology is significantly more modest. The reasons lie in the considerations for mission-ready rovers stated above, along with the difficulty in assembling an integrated rover architecture from a number of disparate, individual research results.

Mission-ready rovers must operate in environments that are largely unknown and unstructured. Many indoor robots rely on an accurate map of the environment. Furthermore, the environment is often assumed to have straight walls, flat floors, and right angles at corners. These assumptions are obviously not true for outdoor environments, and the approaches are not easily extended to the general case.

Map learning has been proposed as a method for overcoming the lack of an accurate map [9]. However, this is difficult in unstructured environments with significant motion error. Moreover, the usefulness of a map-learning phase is dubious in an environment where exploration and science tasks are foremost; the extra time necessary to build accurate navigation maps will have a severe impact on the overall science return.

Many architectures have been proposed for autonomous robots, relying largely on artificial intelligence techniques for merging high-level and reactive control of the rover [10,11,12,13,14]. Very few architectures address the issues arising in remote, planetary rovers [15,16,17]. In general, the architectures fail in the areas of understandability or safety. The more complex AI techniques that promise higher levels of autonomous operations often come with no guarantee on behavior, and their operation is not easily predictable. Bottom-up approaches that rely heavily on programming and interacting behaviors [18,19] become equally unpredictable when the system becomes large.

These architectures assume tightly coupled interaction between different elements of the architecture (integrated planning and execution) or between a human and the rover system (teleoperation). Since it is not currently feasible to put a full planner on board a planetary rover, either option fails the requirement of limited communication.

Research rovers, both indoors and out, are tested repeatedly under similar environments and situations to demonstrate the usefulness of the rover architectures under those conditions. Based on the test results, models are refined, parameters adjusted, and the rover control software modified. However, the only environment that truly matches that of a planetary mission is on the planet. Once there, the rover has a limited lifetime, which must be devoted to mission goals, leaving no time for further testing and refinement.

Rover research technology often addresses one problem, such as navigation, in isolation. A deployed rover must, however, balance considerations such as navigation with science, communication, resource consumption, and fault recovery. An integrated system presents challenges that single technologies do not address. Nonetheless, the research literature contains many results that are of potential value to missions. Some have influenced current rover designs, but many more remain unused. Individual technologies, in addition to those mentioned above, include verifiable real-time control [20,21,22] and human-computer interaction [23].

The next generation of mission-ready rovers

The next generation of rovers, the first of which is expected to launch in 2003, will be more flexible than Sojourner. Instead of simple command sequences, the rovers will execute complex contingency plans, which tell the rover explicitly what to do if something goes wrong. They will also execute plans more robustly, so minor problems such as incorrect resource estimates or motor overheating do not cause complete failure. Finally, they will be able to identify and diagnose internal faults and recover from simple failures.

To imagine how these rovers will behave, consider again the problem of backing up to a rock, deploying a sensor, gathering data, and moving on. We can imagine what a smart rover would need to do in this case. First of all, the operation of backing up on Sojourner was brittle because it used a simple “try three times” strategy. We can imagine a more robust operation of backing up until contact, with some timeout in case it hits an insurmountable difficulty. The backing-up operation should include the ability to try different approach paths if obstacles block the planned route. The rover should also be able to take a moment to let an overheating motor cool down rather than abandoning the operation. In addition, the rover should notice that a wheel seems to be malfunctioning and pulling the rover off-path,

and it should autonomously shift control algorithms to compensate. Certainly the rover should verify correct instrument placement before doing hours of measurements; furthermore, the rover should have alternative plans in case it cannot make contact with the rock despite its best efforts. While it is performing measurements, the rover should monitor its energy level to make sure that it will have enough energy left to send its data to Earth during the next communication event. It should cut short the measurements if it ends up in the shade of a larger rock and cannot charge its battery enough to complete the task and also communicate.

The technology needed to construct such a rover is within reach using artificial intelligence technology in development today.

3. Architecture Overview

We now discuss our rover autonomy architecture, which is specifically designed to accommodate the particular constraints of planetary rovers while supporting autonomous operations. The architecture builds on elements of the Remote Agent architecture [24], extending and modifying existing elements and adding new elements as needed.

The Remote Agent has been applied to the problem of spacecraft control, in particular a technology experiment on the Deep Space One mission [25]. The differences between spacecraft in the relatively predictable environment of space and a rover interacting with an unknown planetary surface have led to the particular elements of the current architecture.

The rover autonomy architecture consists of four major reasoning components (see Figure 1): a contingency planner/scheduler (CPS), a conditional executive (CX), a resource manager (RM), and a model-based mode identification and reconfiguration system (MIR). In the current system, CPS is a ground-based planner, which is given high-level science goals and generates a temporally flexible schedule along with contingency plans for possible execution failures. The planning capabilities will incrementally migrate on board as advances in time-constrained planning and mission-qualified processor power allow and as the need for autonomy increases. The contingent schedule is refined with help from a rover operator and then sent to the on-board executive CX. These commands are sent to the real-time control system, with results coming back via state monitors into MIR. MIR's *mode identification* layer infers the system state from the monitored information and updates the state for CX. If commands fail or schedule constraints are violated, CX tries to recover using the contingency plans. In the future, CX will also be able to call MIR's *mode reconfiguration* layer to produce a recovery plan for unanticipated failures.

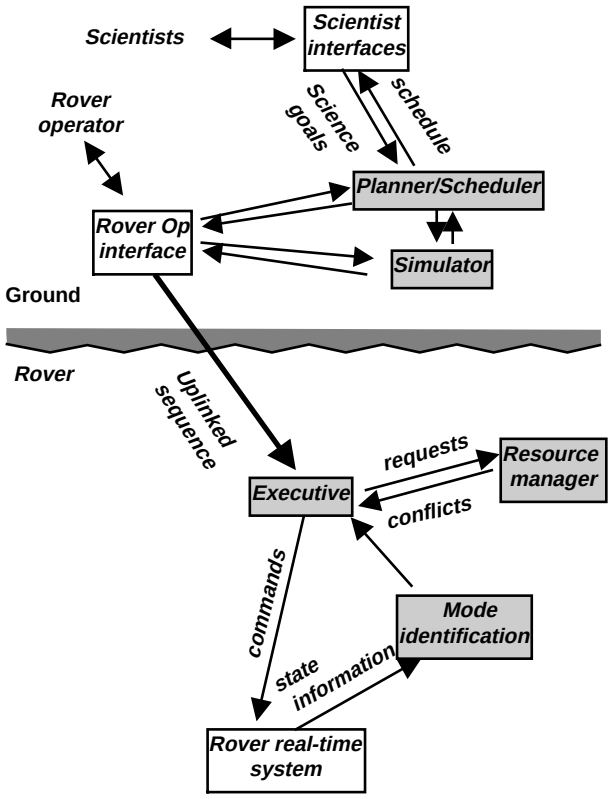

Figure 1: Rover autonomy architecture

Contingency Planner/Scheduler

Throughout a mission, detailed mission operations plans must be constructed, validated, and uplinked to a spacecraft or rover. Currently a mission operations plan takes the form of a rigid, time-stamped sequence of low-level commands. Unfortunately, there is uncertainty about many aspects of task 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, current plans are based on worst-case estimates and contain fail-safe checks. If tasks take less time than expected, the spacecraft or rover just waits for the next time-stamped task. If tasks take longer than expected, they may be terminated before completion. In fact, all non-essential operations may be halted until a new command sequence is received. All of these situations result in unnecessary delays and lost science opportunities.

To account for execution uncertainty, CPS can actively plan for, and take advantage of, possible contingencies. Thus, if an operation takes longer than a certain amount of time, or the power remaining drops below a specified value, a different pre-planned sequence of operations can be performed. Building contingency plans is, in general, intractable, and so contingency planners tend to be slow [26,27,28]. To overcome this problem, CPS employs the *Just-in-Case* (JIC) planning approach [29], originally

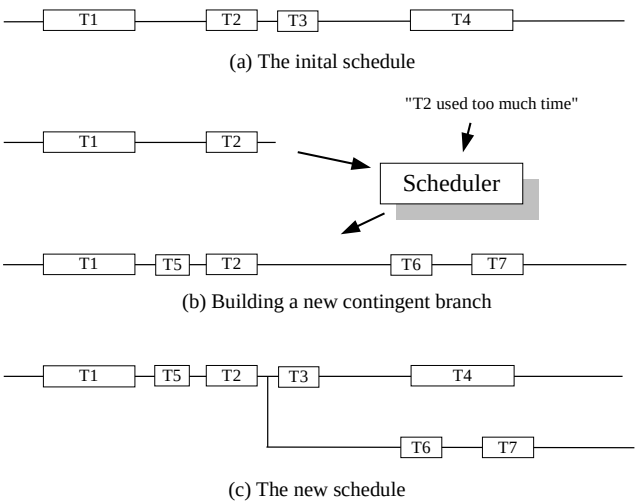

Figure 2: The scheduling process. Initially, the scheduler builds a straight-line schedule assuming expected outcomes (a). The scheduler then finds a likely breakpoint; in the example, the task T2 is found to often take too long, preventing T3 from executing. The schedule up to the offending task, plus the failure condition, is fed back into the scheduler (b). The output is a new straight-line schedule that is the contingent alternative. The original and new schedules are then merged back into one (c), with an explicit test of the contingency added in after task T2.

developed to generate contingent observation schedules for automated telescopes.

The basic idea of JIC is to take an existing schedule and look for the places where it is most likely to fail. The JIC scheduler then generates alternative schedules for each of those situations. The JIC scheduler starts with a sequence of tasks, where each task must be performed in a certain temporal window. However, there is uncertainty in how long a particular task may take, and this can lead to potential failures of the schedule. For example, an execution failure could result if one task finished sufficiently late that the next task's start window has already passed.

CPS operates at two levels. At the lower level, CPS builds straight-line (non-contingent) schedules of the tasks that it is given. It does this by piecing together a schedule, one task at a time. CPS uses a local search strategy to determine which task should be added to the schedule next, and where. The local search strategy has the advantage of being an anytime algorithm—over time, CPS will produce schedules of increasing quality.

At the higher level, CPS actually builds the contingent schedule using this lower level as a subroutine. An initial straight-line schedule is built first, then contingent branches are iteratively added (see Figure 2). In each iteration, the point in the contingent schedule that is most likely to fail is selected. Then, the low-level scheduler is called to build a new straight-line schedule, given the breaking condition and the schedule prefix up until the breakpoint as an initial seed. The resulting schedule is then integrated into the existing contingent schedule and the iteration is complete.

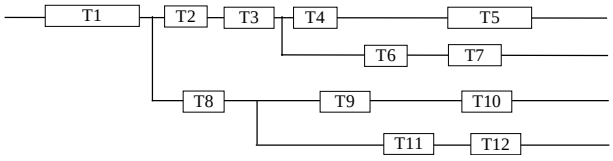

Figure 3. A plan with a nominal sequence (top) and a set of contingent branches.

The initial JIC work dealt only with one resource: time. For rover operations, uncertainty about other resources can also lead to potential failures in a plan. For example, if a task uses more power than expected, or the battery has not charged as much as expected, there may be insufficient power available for the subsequent tasks. However, there may be enough power left to do other useful tasks. Thus, we have extended JIC to also consider power consumption and data production when it is searching for a probable break point. Furthermore, we consider both when a resource is overdrawn (a task takes too long) and when a resource is available in surplus (a task required less power than expected). This ability allows the scheduler to build contingent plans that take advantage of unexpected surplus.

Conditional Executive

The conditional executive (CX) is responsible for interpreting the command sequence coming from ground control, checking run-time resource requirements and availability, monitoring plan execution, and potentially selecting alternative plan branches if the situation changes.

The input to CX consists of the primary plan and a set of alternate plans. The primary plan contains a nominal sequence and a set of contingent branches (see Figure 3). The nominal sequence is the sequence that will be executed if there are no deviations from the *a priori* expectations of the environment and actions. The contingent branches specify alternative courses of action. Within any contingent branch there may be further contingent branches, hence the primary plan is a tree of alternative courses of action.

The alternate plans are not attached to particular points in the primary plan but are rather applicable at any time their conditions are satisfied; in some sense they are global contingent branches, whereas the contingent branches within a plan are local to their position in the plan. When enabled, each alternate plan can either replace the executing plan or insert itself between actions of the executing plan. Enabling events can include unexpected opportunities, plan failures, or conditions such as resource shortfalls and component degradation.

CX starts by executing the nominal sequence of the primary plan. At each point in time, CX may have multiple options,

corresponding to the eligible branches of a branch point and the enabled alternate sequences. CX chooses the option with the highest estimated expected utility, computed over the remainder of the plan. The utility of successfully completing an atomic action is set by operators on the ground. From this atomic utility and a model of the probabilities of various events (such as a traverse taking longer than anticipated), the expected utility of an entire branching sequence can be estimated. This expected utility is initially computed by the ground planner CPS; the utility could be updated by an on-board plan revision component at run time to reflect changes in resource availability, system state, or the environment.

CX receives state information from the mode identification module MIR and resource information from the resource manager RM. It uses this information to check action preconditions and maintenance conditions, as well as to check the preconditions of the alternate plan library. The ability to branch on any state or resource condition provides the sequence writer with a powerful language for describing activities.

CX extends its own robust activity mechanism using the fault recovery mechanisms of MIR. CX may request recovery commands from MIR and integrate those commands into the procedure for the currently executing action. If MIR cannot suggest a recovery, the action has failed, and depending on instructions in the plan, CX either ignores the failure or aborts the executing plan and checks for applicable alternate plans. In the case that no alternate plans apply, CX aborts the entire plan set and puts the rover into a stable standby mode.

As an example of flexible execution, consider the following plan for a day's traverse: the rover is to the south of a small ridge, trying to head generally north. The uplinked primary plan specifies the following course of action:

- Travel north to the top of the ridge
- Choose between the options:
    - Nominal option, highest utility (precondition: there must be a path)
        - Continue to the north
        - Downlink to ground at sundown
    - Contingent option, lower utility
        - Move back down the ridge
        - Travel east scanning for a pass
        - Downlink to ground at sundown

Because operators have a good view of the slope of the ridge, they decide to precisely define the low-level navigation for that segment. They can upload a time-stamped set of motion commands as a sequence. In this case, CX is operating like a traditional sequencer.

At the top of the ridge, the rover's on-board navigation system judges that there is no safe path to the north. The nominal plan option is not eligible to execute because its precondition has failed, so the contingent option is selected.

As the rover executes the procedure for its action to move back down the ridge, the mode identification component of MIR notices an inconsistency between the commanded speed of the left front wheel and the current draw of its motor, and reports the anomaly to CX. CX invokes the procedure's predefined exception handler for that anomaly, which in turn requests a recovery program from the mode reconfiguration component of MIR. The recovery program resets the motor microcontroller and resolves the anomaly. The entire MIR cycle has happened within the scope of a single action of the plan, below the level of abstraction of the off-board planning system CPS.

Returning to the bottom of the ridge, the rover continues with its uplinked sequence by beginning its traverse to the east. An hour before sundown, the vision system picks up an unusual green patch on a nearby rock. The ground operators had supplied an alternate plan for this serendipitous science opportunity and had given it a high priority. After considering the relative utility estimates for the options of continuing the current plan or inserting the alternate plan to examine the rock before the next action, CX chooses to examine the rock.

After sidetracking to the rock, CX returns to the primary plan just before sundown. The next two actions, "Travel east" and "Downlink," both make requests for time and power from the resource manager RM. It turns out that because of the sidetrack, the total energy requested by these actions exceeds the battery energy stored during the day. RM reports the conflict to CX. The ground operators had given a very high priority to the "Downlink" action and specified that it should run even if previous actions failed, so CX chooses to cancel the lower utility "Travel east" action.

After the "Travel east" command has been cancelled, CX is ready to call the downlink action—but since the plan requires that the downlink begin at a specific time, CX waits until the beginning of that window to execute the downlink. Ground operators can also request that CX wait for an arbitrary predicate to become true, such as the antenna being turned on, before executing an action.

Resource Management

Resources on rovers are severely limited and at the same time critical for mission success. Solar energy is the primary source of power, but downlink events may be scheduled when there is little or no sunlight, so the power must be managed such that the on-board battery is sufficiently charged to communicate. There are more opportunities to take pictures and instrument measurements than there is space on board or communication bandwidth, thus space needs to be managed so that the most important data are stored and sent back.

Command sequences are sent at periodic intervals, usually daily. Sequence writers (or automatic planning and scheduling tools) must make conservative estimates of the resource usage to avoid oversubscribing the resource (for example, draining the batteries). But overly conservative

estimates may not make full use of the resources, leaving the rover idle or passing up science opportunities that were in fact possible. Conversely, overly optimistic estimates may lead to fault conditions that will at best break the plan and at worst damage the rover.

The underlying problem is that resources cannot be estimated precisely for an entire day since the rover's interactions with the environment are complex. Only during the execution of the sequence will it become clear how much of each resource is indeed available. If a traverse completes quickly because of unexpectedly easy terrain or good traction, more tasks may be possible with the surplus time and power. Conversely, if the rover traverses a ridge, slanted away from the sun, the accumulated energy may be insufficient to run all the planned experiments and also communicate with ground. The rover would then have to discard some of the experiments to reserve energy for communication.

We have designed an on-board, run-time resource manager that receives estimated resource profile information from tasks, monitors current and planned resource usage, and reacts to changes in resource availability. The resource manager is largely transparent to sequence writers, while allowing them to take full advantage of the resources available. The following are the primary contributions of the resource manager:

- **On-board resource conflict and opportunity detection**. The resource manager is able to notice when there are differences between predicted and available resources, both in the present and the future. A conditional sequencer that branches on resource conditions can take advantage of changes and run the plan that best conforms to the known resource information.

- **On-board resource conflict recovery**. The resource manager has multiple possible strategies for recovering from resource conflicts. The strategies include rejecting the conflicting tasks (the simplest and most severe strategy) or shedding low-priority tasks.

- **On-board resource opportunity exploitation**. The resource manager is able to allow low-priority or background tasks (like heating of the electronic components) to take advantage of unexpected extra resources. In conjunction with a conditional sequencer, resource opportunities can trigger new plan fragments that wait for this additional resource availability.

Note that detection, recovery, and opportunity exploitation can operate on future resource requirements and predicted availability. This will provide support for more flexible task scheduling and, in the long term, for further on-board autonomy such as automatic scheduling, sequence generation, and fault recovery.

Interaction between resource management and sequence execution—The resource manager RM in the rover autonomy architecture communicates with the sequence execution component CX. Each step in the nominal sequence has an expected resource profile associated with it. The ground-based planner/scheduler CPS uses the expected profiles and the expected resource availability to construct the sequence. Under normal conditions, the initial sequence will respect the resource availability (although this is not assumed, in case the resource availability changes before uplink).

CX sends the expected profiles to RM, which records them and checks for conflicts (see Figure 4). Any conflicts are signalled to CX. CX in turn can respond in a variety of ways, depending on the severity and immediacy of the conflict. CX can fail the plan, select an alternate plan from its plan library, or ignore the conflict (for example, in the case that it is a future conflict with a low priority task).

As the sequence is executing, the estimates of resource usage and availability become concrete. Resource monitors gather the information about the real usage and availability and send that to RM. Based on this new information, conflicts or opportunities may arise, which RM will in turn signal to CX.

Conflict detection and recovery—RM stores resource profiles using a timeline-based representation. Each timeline is a set of non-intersecting time intervals over which a constant amount of the resource is used (see Figure 5).

The predicted resource availability, given by system resource information and models of resource availability

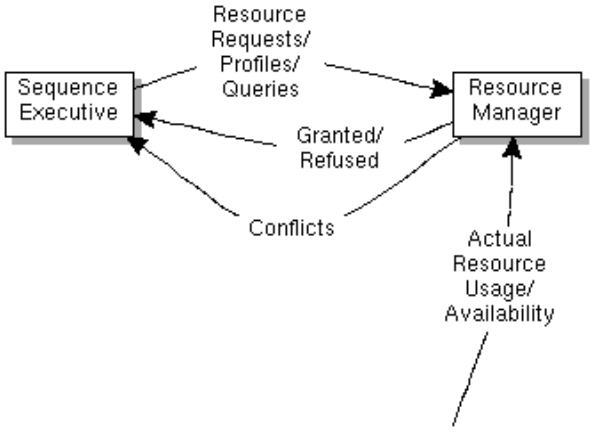

Figure 4: Communication between resource manager and sequence executive.

over time (for example, solar flux models), defines a timeline of the maximum available resource during any time interval.

Each resource request includes a profile of predicted resource usage. The sum of all the granted resource requests itself defines a timeline, and this resource request timeline

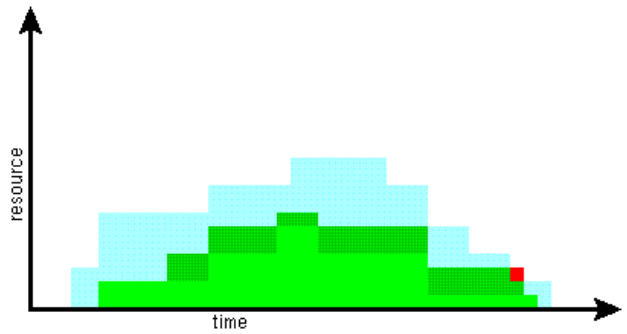

Figure 5: Resource management timelines. Resource availability is shown in the lighter shade, usage in darker shades. Conflicts occur where actual or predicted usage exceeds availability.

must be less than the resource availability timeline at all time points.

A conflict will arise whenever the request timeline exceeds the available timeline. This may occur because of availability changes or because of a new request. In either case, the set of tasks using that resource during the time interval where the conflict occurs forms the *conflict set*. This conflict set is then minimized to find a set of lowest-priority tasks that, when removed, will resolve the conflict. In the case of a new request, the tasks must be lower priority than the new task; if the conflict cannot otherwise be resolved, the request is refused, thus resolving the conflict. If a minimal conflict set is found that will resolve the conflict, RM sends that set to the executive CX, which will react to the conflict. For example, CX might remove those tasks from its planned sequence, potentially even aborting the sequence if those tasks were necessary for sequence success.

Resource borrowing—RM has the ability to borrow resources from tasks to satisfy new resource requests. For tasks that can operate in a number of modes, or background tasks that can be preempted, the resource profile includes an indication of how much of the resource requested could be given up without aborting the task. For example, an image could be compressed with quality loss to free up data storage for higher priority images, but without completely discarding the image. RM prefers to borrow resources before it sheds tasks.

Diagnosis and Recovery

The approach that Sojourner used to respond to faults is fairly typical of most fault-protection systems: monitors on specific sensors check whether given limits have been exceeded. If so, the rover stops what it's doing and waits for the next plan uplinked from Earth. Such monitors look at pitch and roll sensors, accelerometers, and the product of current and time in motors. This approach has a number of limitations.

- The appropriate thresholds for a given sensor depend greatly on context, not the least of which is the health condition of the sensor itself. Wheel motors will draw more current when the rover is going uphill, for instance. In the Pathfinder mission, the ground operations team turned off specific accelerometers during particular maneuvers, and changed variables that influence the rover's reaction to minor faults. While this is one solution to the context dependence of sensor data, it is a fairly limited solution.

- Given that appropriate sensor thresholds are context-dependent, rover engineers must trade the cost of false alarms against the risk of not responding to failures in time to prevent damage. On Sojourner, false alarms were quite common, resulting in lost science opportunities.

- Fault protection systems that rely only on monitoring individual sensors cannot cope well with multiple faults, and they are limited in the number of faults they can detect.

- Simply halting and calling home for help is unnecessarily conservative for many common faults. Often, the rover could recover from the failure and keep going or else perform actions that do not depend on (or affect) the damaged component. Doing so depends on knowing the nature of the fault.

To address these problems, the rover autonomy architecture makes use of diagnosis, using all available sensor data to infer whether the rover is behaving correctly and, in some cases, to infer the specific fault. When it is possible to infer the fault, and that fault is recoverable, the rover can execute the appropriate recovery plan; otherwise, it can always shut down and call home for help. Diagnosis, taking into account the complete state of the rover, allows faults to be detected earlier and also reduces the number of false alarms.

We refer to the diagnosis and recovery component of the rover autonomy architecture as MIR, for Mode

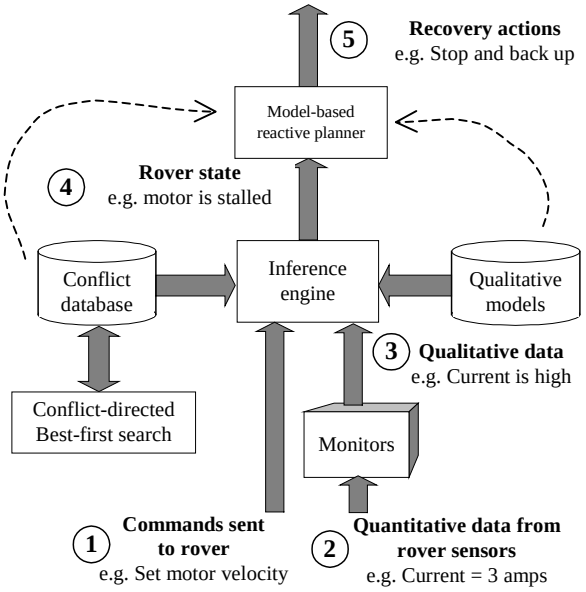

Figure 6: Information flow in MIR

Identification and Reconfiguration. MIR is part of the Remote Agent Experiment flying on board the DS1 spacecraft [24]. MIR is a discrete, model-based controller that uses a single, declarative model of the rover for both mode identification (inferring the internal state) and reconfiguration [30] (changing the system to a more desirable state). Like CX, MIR runs as a concurrent reactive process. MIR itself contains two components, one for Mode Identification (MI) and one for Mode Reconfiguration (MR).

MI is the sensing component of MIR's model-based reconfiguration capability. MI 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.

MR serves as a recovery expert to CX, taking as input a recovery request and returning a sequence of operations that, when executed starting in the current state, will move the executive into a state satisfying the properties required for successful execution of the failed activity.

These recovery actions are determined by a model-based reactive planner (see Figure 6), called Burton [31], which uses a universal plan compiled from the models to quickly determine the next action to execute, based on the current state and the recovery request. The time required to generate a complete recovery plan is linear in the length of the plan.

MIR uses algorithms adapted from model-based diagnosis [35,32] to provide the above functions. MIR extends the basic ideas of model-based diagnosis by modeling each component as a finite state machine (see Figure 7), and modeling the whole rover as a set of concurrent, synchronous state machines. These finite-state-machine models are used to predict state changes resulting from executive commands and to identify commands for recovery that will take the rover to the desired state. Because the rover is modeled as a concurrent state machine, MI can track concurrent changes to the different components of the rover. This methodology is independent of the actual set of available sensors and commands. Furthermore, it does not require that all aspects of the rover state are directly observable, providing an elegant solution to the problem of limited observability. Following [32], MIR uses a conflict directed best-first search in MI, to find the most likely set of state assignments for all components, and in MR, to find the least-cost sequence of commands needed to achieve the desired recovery (see Figure 6).

The use of model-based diagnosis algorithms provides a number of advantages including sound and complete search algorithms for MI and MR, the ability to handle multiple faults, and the ability to reason with explicit fault models if they are available. Alternatively, MIR can infer faults solely on the basis of a model of nominal behavior.

Additionally, the models are modular, which simplifies writing and maintaining them and aids reuse. For example, although the Marsokhod has six wheels, each containing a motor, only one wheel module is needed; furthermore, the module is general enough that it also applies to rovers with different wheel configurations, including Carnegie Mellon University's Nomad rover.

The behavior of each component state is captured using abstract, qualitative, models [33,34], which describe qualities of the rover's structure or behavior without the detail needed for precise numerical prediction. Abstract models are much easier to acquire and verify than quantitative engineering models, and are easier to reuse. For example, the wheels on the Marsokhod and Nomad rovers are quantitatively quite different, but they are qualitatively similar, and thus the wheel module shown in Figure 8 can be used for both rovers (though the motor modules differ slightly, since Nomad uses brushless motors). The same applies to the majority of components.

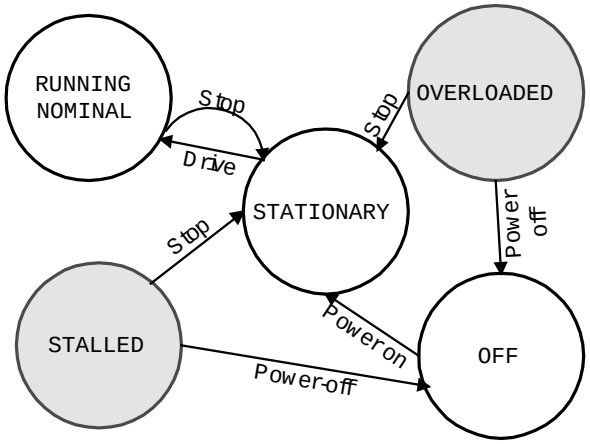

Figure 7: State diagram for motor control component. Shaded nodes represent fault modes. In addition to the explicit transitions, indicated by arrows, there are also implicit, random transitions to fault nodes, represented in the model probabilistically.

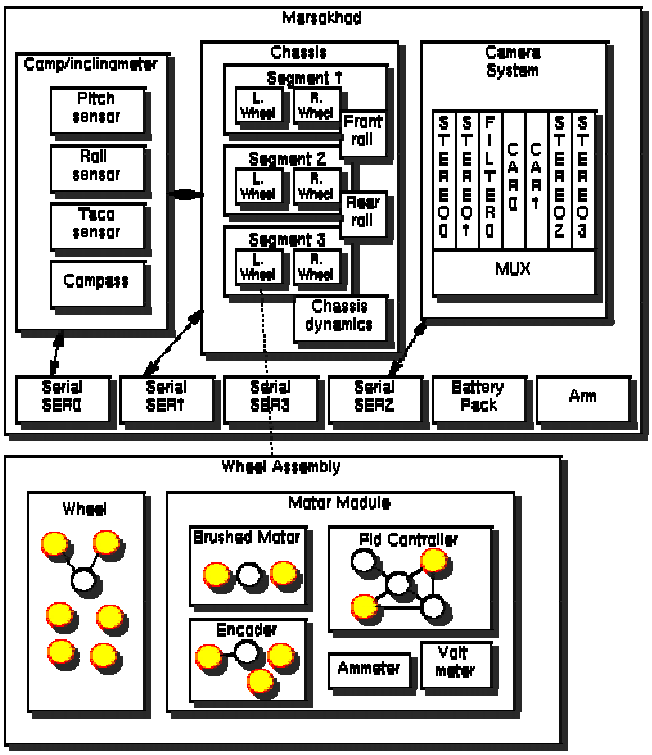

Figure 8: Partial MIR model of Marsokhod rover. Nested boxes indicate sub-modules. Boxes containing state diagrams are the lowest-level components; the nodes of the state diagrams are the possible states, or modes, of the component. Arcs between nodes indicate explicit, deterministic, transitions between nodes. Implicit, random, transitions to failure modes (shaded circles) are also possible. The box labeled "Wheel Assembly" shows a more detailed description of the module used to represent the six wheels on the rover.

While such models cannot specify how far to the left the rover will drift if the motor for one of its left wheels has degraded, it can be used to identify the source of failure, given evidence of the compass, encoders and ammeters. 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.

4. FIELD TEST

The rover autonomy architecture will be demonstrated as part of a February 1999 field test using the NASA Ames Marsokhod rover [5]. The field test is designed to expose scientists to new technology that is intended for use in near-term missions, and to give them a chance to interact with these technologies in a mission-realistic setting.

The role of the rover autonomy architecture will be to enable reliable operation of the rover during the field test and to illustrate the contribution of the rover autonomy elements to the science productivity of the rover. Part of the field test will be dedicated to technology experiments, and the rover autonomy architecture will be fully exercised during that time. Otherwise it will be simply functioning as an integral part of the overall rover system.

The Marsokhod rover

The 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 and an arm equipped with a spectrometer and cameras. The on-board computing environment is a Pentium-based Linux system, for ease of research software integration.

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. The field tests were designed to study user interface issues and science instrument selection. This field test will have the added capabilities provided by the rover autonomy architecture.

Objectives for the rover autonomy architecture

The field test will demonstrate the rover autonomy architecture both in specific technology experiments and as part of the integrated rover system during science experiments. We expect to demonstrate and test particular aspects of the rover autonomy architecture in this field test:

- **Description level**. The language used to communicate with the ground system, as well as the uplinked sequence language, are designed to be at an appropriate level of description for the user of that component. The scientists are expected to interact with the rover autonomy system at the level of goals and, in the case of the principal investigator (PI), conditional schedules. The uplinked sequence language is designed to capture enough detail to describe the day's operation without losing the flexibility and conditionality that will increase science productivity and reliability**.**

- **Contingency planning and scheduling**. The science PI will be able to specify contingencies under which the science plan will change. In addition, the rover operators will be able to add contingencies to handle rover operational failures such as traversal failure, instrument deployment failures, or predicted resource conflicts. In addition, the science team will be able to add opportunistic sequences to be executed when more resources are available than originally predicted. The contingency planning and scheduling system CPS will be able to automatically generate contingency branches at the most likely failure points in the schedule, add or delete branch points with user direction, and verify that

schedules respect resource and time constraints. CPS will generate a conditional sequence as its output.

- **Conditional execution of flexible plans**. The rover will be able to execute plans with temporal flexibility and conditional branches. Task decomposition from a higher-level sequence to rover-level commands will be performed interactively during this field test rather than automatically on board the rover. The conditional execution system CX will accept the rover-level sequence, with contingency branches and additional plan fragments in the plan library, and execute the sequence until successful completion or failure.

- **Fault identification**. The rover will be able to identify errors that may not yet exceed hard thresholds, but should nonetheless be handled by modifying or aborting the sequence. In this field test the sensing component MIR will be passive, based on information supplied by the executive (which commands are executed) and the rover control system via monitors. During the technology experiment, fault identification will be tested directly; during the science experiment, this will be tested as fault conditions arise.

- **Resource management**. The resource manager RM in the field test will manage power and data storage resources. The rover will monitor these resources and indicate when more or less resources are available than predicted. This will be exercised fully in the technology experiment phase, and then will be used as needed in the science phase.

5. FUTURE WORK

Task decomposition

Each of the plan actions is potentially a high-level command that may be decomposed into a procedure of low-level commands for the rover real-time control system. This procedure can use constructs for robustly executing commands, including catching failure conditions and performing local recoveries and retries. Thus low-level conditionality and looping behaviors may be represented at a level below the granularity of the planner, reducing the computational complexity of the planning problem while guaranteeing the correct execution semantics. For example, the seemingly simple action "move to a point 5 meters straight ahead" may include the ability to pause if a motor overheats, try multiple routes if obstacles block the path, and switch to alternative control algorithms if a wheel becomes blocked.

The ability to decompose actions using a rich procedural representation is a key point of the Remote Agent architecture and will help in the rover autonomy architecture to provide robust implementations of sequenced actions.

Interaction with the environment

In the past, MIR has been used to model systems, such as spacecraft, in which the environment outside the spacecraft can be effectively ignored. Spacecraft follow known trajectories, free of obstacles, in which the external environment can be reduced to a few simple variables, such as the relative position of the Earth. On rovers, interaction with the environment is central to many of the possible faults: dust accumulates on the solar panels, the rover passes into the shadow of large rocks, or it gets caught on small ones. In order to handle failures involving external factors, we need to add the capability to reason from first principles about the rover and its interaction with the environment. We intend to combine the model-based deduction used by MIR with hybrid simulation, to capture factors such as rover kinematics.

Active sensing and testing—The sensor information that MI can passively acquire is not always adequate to allow an unambiguous diagnosis, which may make it impossible for MR to identify the appropriate recovery actions. In general, it may be ambiguous which of two failures has occurred if both are consistent with observations, and it may be ambiguous whether a given component has failed or the sensor responsible for measuring that component has failed. If an encoder indicates that a wheel drive motor is not turning, or is turning too slowly, that could be a sensor error: the encoder may be skipping counts or entirely dead. While MI could look at other evidence, such as motor currents, to see if the motor appears to be stalled or encountering unusually high torque, that evidence may not be sufficient to determine the true fault. If the encoder is giving low readings, the PID controller, which tries to maintain the desired encoder values, will tend to speed up the motor, resulting in higher wheel currents, which could be taken as evidence of excessive torque. Furthermore, failures such as a motor stall or excessive torque can reflect a number of different underlying faults, such as seized bearings or a rock caught in the wheel.

The true state of the rover may be determined by performing experiments designed to eliminate certain candidate diagnoses. For example, if the wheel does not appear to be turning, the rover could try backing up to see if the wheel is caught on a rock. Or it could try driving with some subset of the wheels, and use other sensors to determine if the rover is moving in a manner consistent with the encoder readings.

To deal with these ambiguities, we are adding the capability to perform active sensing and testing in order to narrow the candidate situation assessments (diagnoses) and in order to evaluate the utility of alternative recovery plans.

In support of this active testing, MIR can make use of its models, both to determine when there are multiple competing diagnosis and to identify activities it can perform that will rule out or confirm certain hypotheses. Reasoning about the information to be gained by executing actions exceeds the ability of the MIR system designed for the

Remote Agent, but we are working to provide that capability.

Work in active testing for diagnosis [35] is typically based on probe selection for circuit diagnosis, and it relies on certain simplifying assumptions that are valid for circuits but not for rovers. Some of the key assumptions are:

1. Measurements do not affect the state of the system being diagnosed.
2. All measurements have equal cost.
3. The goal of making measurements is to eliminate ambiguity as quickly as possible (i.e., to minimize the total number of measurements); the order of measurements is otherwise irrelevant.

These assumptions lead to a *minimum entropy* measure for probe selection. The next probe selected is the one that results in the lowest expected entropy of the probability distribution of diagnoses. This policy tends to minimize the total cost of measurements, under the assumptions listed above. However, these assumptions do not hold in the rover domain, for the following reasons, so minimizing entropy is not sufficient.

1. Any information that can be obtained without changing the state of the rover, as long as it is not too expensive to compute, is already continuously available to MI. Any additional tests involve causal action, such as spinning a wheel or taking a picture from a camera.
2. On a rover, some sensing actions may have very high cost, including the possibility of causing some undesirable side effects, while others are relatively cheap.
3. In the rover autonomy architecture, the main purpose of diagnosis is to disambiguate the rover state enough to find an appropriate recovery plan. Thus, not all ambiguities are equal: the value of information depends on the value of the recovery it supports. In the case of multiple faults, one fault may be more critical and need immediate response, meaning measurements relevant to that fault have priority. If several candidate faults have the same recovery procedure, fully disambiguating the fault may even be unimportant.

We are exploring a modification of the minimum-entropy model, which ranks measurements according to the recovery actions they support and penalizes measurements based on the cost of the corresponding sensing actions.

6. Conclusions

The particular characteristics of Mars rover operations require a significant level of rover autonomy and an ability to handle resource constraints and unpredictable events. We have designed an integrated architecture for rovers that includes contingency planning on ground and flexible, robust execution of conditional sequences on board. The on-board executive draws on model-based fault diagnosis, active sensing, and dynamic resource management to maximize its science return.

In the future, we would like to see rovers that are capable of even higher levels of autonomous operations. These rovers will accept very high-level goals from human operators and will be able to achieve those goals with no further supervision, even in dynamic and uncertain environments. These rovers will be self-diagnosing and self-repairing; they will be capable of detecting gradual degradation, adjusting internal parameters accordingly, and performing preventive maintenance to avoid catastrophic failure. For example, solar panels accumulate dust over time and are gradually damaged by UV; the ability of the rover to execute a plan will be dependent on its gradually decreasing energy production, and it may need to perform actions to remove dust periodically when operating over long time intervals. In addition to expected and predictable degradation, rovers will be able to automatically replan when unexpected problems occur or serendipitous opportunities arise.

We are building toward this vision in our research on robust autonomous rovers. While we are not there yet, it is reasonable to expect such capabilities in future generations of rovers.

7. Acknowledgements

We would like to thank the following people for their contributions:

Michael Sims for giving us the opportunity to participate in the Marsokhod field test and for his very helpful comments on this paper. Hans Thomas and the rest of the Intelligent Mechanisms Group at NASA Ames Research Center for helping us integrate our architecture with the Marsokhod user interface and control software.

Katherine Smith for her ongoing work to implement MIR monitors and models. Barney Pell for his contributions to the initial resource management ideas. David Miller, the IMG, and the Exec team for their involvement in the discussions defining the SCS language. Jim Kurien and Chris Plaunt for their help in understanding and reusing existing Remote Agent code.

References

[1] A. H. Mishkin, J. C. Morrison, T. T. Nguyen, H. W. Stone, B. K. Cooper, and B. H. Wilcox, “Experiences with Operations and Autonomy of the Mars Pathfinder Microrover,” Proceedings of the IEEE Aerospace Conference, 1998.

[2] S. Hayati and R. Arvidson, “Long range science rover (Rocky 7) Mojave Desert field tests,” i-SAIRAS, 1997.

[3] R. Arvidson and S. Hayati. Mojave field experiments for Rocky 7 prototype Mars rover. URL: http://wundow.wustl.edu/rocky7.

[4] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W. Whittaker. “Ambler: An autonomous rover for planetary exploration.” IEEE Computer, June

1989.
[5] D. Christian, D. Wettergreen, M. Bualat, K. Schwehr, D. Tucker, and E. Zbinden. “Field experiments with the Ames Marsokhod rover.” In Proceedings of the 1997 Field and Service Robotics Conference, December 1997.
[6] D. Wettergreen, C.Thorpe, and W. Whittaker, “Exploring Mount Erebus by walking robot,” Robotics and Autonomous Systems, 11(3-4), pp. 171-185, December 1993.
[7] D. Bapna, E.Rollins, J. Murphy, E. Maimone, W. Whittaker, and D. Wettergreen, “The Atacama Desert trek: Outcomes,” IEEE International Conference on Robotics and Automation, pp. 597-604, May 1998.
[8] T. A. Linden and J. Glickman, “Contingency Planning for an Autonomous Land Vehicle,” Proceedings of IJCAI-87, 1987.
[9] S. Thrun, D. Fox, and W. Burgard, “Probabilistic mapping of an environment by a mobile robot,” IEEE International Conference on Robotics and Automation, pp. 1546-1551, May 1998.
[10] B. Hayes-Roth, K. Pfleger, P. Lalanda, P. Morignot, and M. Balabanovic, “A domain-specific software architecture for adaptive intelligent systems,” IEEE Transactions on Software Engineering, 21(4), pp. 288-301, April 1995.
[11] D. E. Wilkins, K. L. Myers, J. D. Lowrance, and L. P. Wesley, “Planning and Reacting in Uncertain and Dynamic Environments,” Journal of Experimental and Theoretical AI, 7(1), pp. 1970227, 1995.
[12] R. Simmons, “An architecture for coordinating planning, sensing and action,” in Innovative Approaches to Planning, Scheduling, and Control, pp. 292-297, 1990.
[13] R. Alami, R. Chatila, S. Fleury, M. Ghallab, F. Ingrand, “An architecture for autonomy,” International Journal of Robotic Research, 17(4), pp 315-337, April 1998.
[14] R. P. Bonasso, D. Kortenkamp, D. Miller, and M. Slack. “Experiences with an architecture for intelligent, reactive agents.” In Proceedings of IJCAI-95, 1995.
[15] D. B. Smith and J. R. Matijevic, “A System Architecture for a Planetary Rover.” Proceedings of the NASA Conference on Space Telerobotics, JPL Publication 89-7, 1989.
[16] R. Chatila, S. Lacroix, T. Simeon, and M. Herrb. “Planetary exploration by a mobile robot: mission teleprogramming and autonomous navigation.” Autonomous Robots, 2(4):333--344, 1995.
[17] G. Giralt and L. Boissier, “The French planetary rover VAP: concept and current developments.” Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS ’92), pp. 1391-1398, 1992.
18 Brooks
[19] E. Gat, R. Desai, R. Ivlev, J. Loch, and D. Miller. “Behavior control for robotic exploration of planetary surfaces.” IEEE Transactions on Robotics and Automation, 10(4), August 1994.
[20] J.-J. Borrelly, E. Coste-Manière, B. Espiau, K. Kapellos, R. Pissard-Gibollet, D. Simon, and N. Turro, “The ORCCAD Architecture.” The International Journal of Robotics Research, 17(4), pp. 338-359, April 1998.
[21] S. A. Schneider, V. W. Chen, G. Pardo-Castellote, H.H. Wang, “ControlShell: A Software Architecture for Complex Electromechanical Systems,” The International Journal of Robotics Research, 17(4), pp. 360-380, April 1998.
[22] D. Musliner, E. Durfee, and K. Shin. “Circa: A cooperative, intelligent, real-time control architecture.” IEEE Transactions on Systems, Man, and Cybernetics, 23(6), 1993.
[23] D. C. MacKenzie and R. C. Arkin, “Evaluating the Usability of Robot Programming Toolsets,” The International Journal of Robotics Research, 17(4), pp 381-401, April 1998.
[24] N. Muscettola, P. P. Nayak, B. Pell, and B. C. Williams. “Remote agent: To boldly go where no AI system has gone before.” Artificial Intelligence, 103(1/2), August 1998.
[25] D. E. Bernard, G. A. Dorais, C. Fry, E. B. Gamble Jr., R. Kanefsky, J. Kurien, W. Millar, N. Muscettola, P. P. Nayak, B. Pell, K. Rajan, N. Rouquette, B. Smith, and B. C. Williams, “Design of the remote agent experiment for spacecraft autonomy,” Proceedings of the IEEE Aerospace Conference, Snowmass, CO, 1998.
[26] D. Draper, S. Hanks, and D. Weld. “Probabilistic planning with information gathering and contingent execution.” In Proc. 2nd Intl. Conf. AI Planning Systems, June 1994.
[27] L. Pryor and G. Collins. “Planning for contingencies: A decision-based approach.” J. Artificial Intelligence Research, 1996.
[28] D. S. Weld, C. R. Anderson, and D. E. Smith. “Extending Graphplan to handle uncertainty & sensing actions.” In Proceedings of AAAI-98, pages 897--904, 1998.
[29] M. Drummond, J. Bresina, and K. Swanson. “Just-in-case scheduling.” In Proceedings of the 12th National Conference on Artificial Intelligence, 1994.
[30] B. C. Williams and P. Nayak, “A Model-based Approach to Reactive Self-Configuring Systems,” Proceedings of AAAI-96, 1996.
[31] B. C. Williams and P. P. Nayak (1997).), “A reactive planner for a model-based executive”, *Proceedings of IJCAI-97*.
[32] J. de Kleer and B. C. Williams, “Diagnosis With Behavioral Modes,” Proceedings of IJCAI-89, 1989.
[33] D. S. Weld and J. de Kleer, *Readings in Qualitative Reasoning About Physical Systems*, Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.
[34] J. de Kleer and B. C. Williams, *Artificial Intelligence*, Volume 51, Elsevier, 1991.
[35] J. de Kleer and B. C. Williams, “Diagnosing Multiple Faults,” Artificial Intelligence, Vol 32, Number 1, 1987.

Richard Washington *is Research Scientist in the Computational Sciences Division at NASA Ames Research Center. He holds a Pd.D. in Computer Science from Stanford University. His research interests include planning, plan execution, reasoning under uncertainty and autonomous robots. He is a member of AAAI and ACM.*

Keith Golden *is a Research Scientist in the Computational Sciences Division at NASA Ames Research Center. He holds a Pd.D. in Computer Science from the University of Washington. His research interests include planning, knowledge representation, sensing and information gathering, software agents and rover autonomy. He is a member of AAAI.*

John L. Bresina *is a Senior Research Scientist within the Autonomous Systems Group of the Computational Sciences Division at NASA Ames Research Center. He holds a Ph.D. in Computer Science from Rutgers University, New Brunswick, NJ. Since 1988, he has been carrying out artificial intelligence research at NASA Ames. He currently leads the group's activities on autonomous planetary rovers, and he also leads a project on telescope observation scheduling. His research interests include planning, scheduling, problem reduction, intelligent execution, reactive agents, and optimization search. He is a member of AAAI and ASP.*

David E. Smith *is a Senior Research Scientist in the Computational Sciences Division at NASA Ames Research Center where he is involved in contingency planning for rover operations, planning and scheduling for the Sofia airborne observatory, and more basic research on temporal planning techniques. Prior to joining NASA he was a Senior Scientist at the Rockwell Science Center where his work led to a commercially fielded system for newspaper imposition planning, and the Design Sheet conceptual design system in use throughout Rockwell and Boeing. Smith received B.S. degrees in Electrical Engineering and Physics from Iowa State University in 1978 and received a Ph.D. in Computer Science from Stanford University in 1985. His current research interests include planning, scheduling, search, constraint satisfaction, and optimization. Recently his work has focused on the boundaries between areas, such as temporal planning, planning under uncertainty, constraint satisfaction approaches to planning, and preprocessing techniques for planning problems. He is a member of AAAI.*

Corin Anderson *is a Ph.D. student at the University of Washington. He holds a B.S. and M.S. in Computer Science from the University of Washington His research interests include planning, scheduling, software agents and computer graphics. He was the coach of the first-place team of the 1996 Pacific Regional ACM programming contest. He is a member of AAAI.*

Trey Smith *is a member of the CMU Field Robotics Center. He holds a B.S. in Computer Science from Carnegie Mellon University. He is working on the CMU Mars Autonomy Project. His primary research interests are robotics and artificial intelligence.*