# Continuous Planning and Execution for an Autonomous Rover

**Tara Estlin, Forest Fisher, Daniel Gaines, Caroline Chouinard, Steve Schaffer and Issa Nesnas**

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109-8099
{*firstname.lastname*}@jpl.nasa.gov

**Abstract**

This paper describes how continuous planning and execution techniques can be used to perform intelligent decision-making for an autonomous Mars rover. The resulting system coordinates low-level rover functionality to achieve science objectives while respecting rover resource and operation constraints. It provides capabilities for schedule generation, execution, monitoring, and dynamic modification to recover from unexpected events or failures. To motivate our system framework, we discuss some of the particular challenges we examine to support an autonomous rover. These challenges include properly interacting with rover navigation software, handling uncertainty in state and resource estimations, as well as effectively balancing methods for deliberative and reactive reasoning. We also describe our experiences in testing this work on two JPL rovers, in an effort to demonstrate capabilities that will support future rover missions to Mars and other planets.

## Introduction

NASA's Mars Exploration Program plans to have us visit the red planet over six times in the next two decades. At least four of these missions will involve rovers or other robotic craft that will be used to explore the surface of the planet and perform numerous geological and atmospheric experiments. In order to collect a high volume of science data, rovers will require capabilities for long-range traverses and autonomous operation. A key aspect of these capabilities is the generation and execution of rover command sequences. These sequences specify an ordered list of commands that achieve desired science goals while ensuring no rover operation or resource constraints are violated. Sequences must often be changed or enhanced during execution in response to changing science goals or unexpected conditions. The model of rover operations used for the 1998 Mars-Pathfinder rover and planned for the 2003 Mars Exploration Rovers (MER) is to generate sequences on the ground based on downloaded data describing the rover's state (Mishkin, et al., 1998). If something unexpected happens during sequence execution, such as an out-of-range sensor reading or significant path deviation, the rover will have very limited recovery procedures onboard and will usually be *safed* until further communication from the ground can provide a new command sequence. This procedure often causes hours of lost science time and makes it very difficult to take advantage of unexpected science opportunities.

To address this problem, AI researchers have been developing several key pieces of software that automatically provide the necessary command sequence for achieving science goals. Planning and scheduling systems (Bresina, et al., 1999; Chien, et al., 2000; Jonsson, et al., 2000) take as input a set of science goals, the current rover state, and a model of rover resource and operation constraints to produce a validated plan of activities. Executive systems (Gat, 1992; Simmons and Apfelbaum, 1998) use rover state information to further expand the plan into a detailed set of commands and dispatch these commands to rover-hardware controllers for execution. Planning and scheduling systems typically focus on goal-driven behavior, which enables a robotic system to produce a plan of actions based on a set of high-level goals and constraints. Executive systems typically focus on event-driven behavior, which enables a robotic system to quickly react to changes in its environment and modify its actions accordingly.

This paper describes an approach for using planning and execution techniques as part of a rover's onboard software to provide autonomous sequencing capabilities. This system is intended to run with little communication with ground. It accepts science and engineering goals and creates a rover command sequence (or plan) that respects relevant constraints, while achieving as many goals as possible. The system executes the produced plan by dispatching commands to the rover's low-level control software and monitoring relevant state information to identify current or potential problems. If problems are detected, the system is designed to recover from those situations by using re-planning techniques to add, move or delete plan activities. Through this work, we have also identified a number of challenges for an onboard planning and execution system to not only produce valid plans, but also promote robust and efficient rover behavior. These challenges include properly interacting with the appropriate rover navigation software, handling uncertainty in state and resource estimations, as well as effectively balancing methods for deliberative and reactive reasoning.

In 2001, we spent a significant amount of time testing our current system on two different rovers in the JPL Mars Yard. We will discuss our scenario design for this testing and give an overview of the results including a discussion of how the system handled major scenario elements. Our main objectives for testing included simulating situations that might arise in future rover missions, (such as the Mars Science Laboratory or MSL mission, planned for launch in 2009), providing feedback on our approach, and identifying future directions that should be investigated.

The rest of this paper is organized in the following manner. First, we discuss some key challenges that we have identified for onboard decision-making software. Next, we present our current system approach and explain how this system fits into a larger rover architecture. We then describe a Mars rover scenario that was used to test our system on rover hardware, and describe how our system performed during that testing. Finally, we discuss issues we are addressing in current and future work, review related work, and present our conclusions.

## Challenges for Onboard Decision Making

Most mobile robot efforts at JPL and NASA have concentrated on building software infrastructure for navigation, manipulation and control. High-level decision making for these efforts, including for the Mars Pathfinder mission, was typically done using very simple execution of linear sequences that were tediously created by ground controllers. For the upcoming 2003 MER mission, there are plans to use a ground-based AI planning and scheduling tool to support science plan creation, however, a command sequence will still be manually generated on the ground and uplinked to the rovers. In these models, when a rover encounters a situation that deviates from its uploaded sequence, the fault protection software may attempt some limited resolution methods. Failing that, the rover enters *safe-mode* and must wait for a new command sequence to be sent from earth. This model of operations results in a significant loss in science return since the rover must remain idle, for hours or days at a time, until new commands are received.

More autonomous rovers have the potential for reducing the need for entering *safe-mode* and, as a result, significantly increasing the science value of a mission. New missions are being considered that will require rovers to support more autonomous endeavors such as long-range traversals, complex science experiments, and longer mission duration. However, autonomy software designers face a number of challenges in providing software to support these types of operations. In this paper, we consider a few key challenges for using planning and execution techniques to provide onboard decision-making capabilities.

To generate its own command sequence for carrying out a set of science goals, the rover will need to reason about a rich model of resource and temporal constraints. For example, it will need to predict power consumption of variable duration activities such as downlinks and traverses, keep track of available power levels, and ensure that generated plans do not exceed power limitations. When resources are over-taxed, the rover should be capable of making science/resource trade-offs in an effort to produce the highest science return. The rover will also require execution and monitoring capabilities to carry out the generated plan on the rover platform. An execution system must be capable of commanding the control software, collecting state updates from sensors, and dealing with activity failures or unexpected events.

Sequence generation for rover surface missions also raises a number of interesting challenges regarding spatial reasoning capabilities. One of the dominating characteristics of rover operations is traverses to designated waypoints and science targets. This element is especially key in future missions that intend to explore large geographic areas. Onboard planning and execution software needs to coordinate with several levels of rover navigation software to generate an efficient and achievable rover plan. This coordination will likely include querying a path planner for route information needed to generate a plan of rover activities, using position estimation values to track rover progress, and correctly modifying the plan when navigation and obstacle avoidance software cause the rover to move off the predicted route.

Another predominant challenge in developing onboard autonomy software is dealing with the inherent uncertainty in predicting rover navigation and science operations. The difficulty is compounded by the tight resource and time constraints that a rover typically faces. At the resource and temporal level, the estimation of items such as power, memory and even activity duration can be highly uncertain. Rover missions are directed at exploring unknown planetary terrains. Requirements for traversing these new terrains are hard to predict. For instance, it is unknown what type of sand consistency a rover will be traversing, which can dramatically affect the required duration and power for a traverse. Similarly, the duration and resource requirements for science operations can vary as well. These variations could be simple, such as a lower then expected image compression ratio, or more complex, such as a drilling operation taking more power and time than originally estimated.

Furthermore, at the state level, the estimation of rover position is often a constant source of error. The Sojourner rover only used dead-reckoning capabilities to estimate rover position, which produced a position error of roughly 5-10% of distance traveled and an average heading drift of 13 degrees per day of traverse (Mishkin, et al., 1998). The MER rovers will use more sophisticated techniques to provide position estimation, including an Inertial Measuring Unit and a Sun camera. However, since these rovers will be traveling significantly longer distances then Sojourner, position estimation error will likely be significant for this mission as well. Since a large part of a rover schedule consists of rover moves to different locations, the onboard autonomy software must use estimations of position to predict the duration and resource

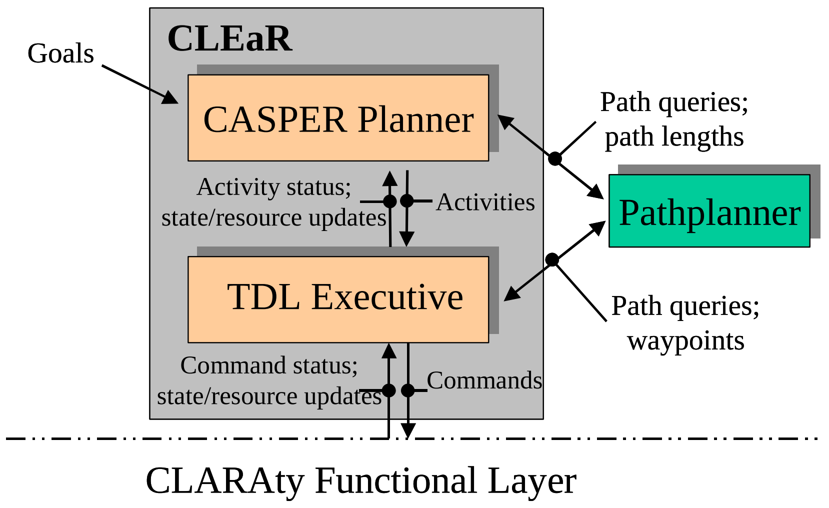

**Figure 1: Planning and Execution System**

requirements of different operations. If these predications are inaccurate, the autonomy software must be able to continuously modify the schedule to handle changes in expected rover behavior.

## A Planning and Execution System for Rover Operations

To address the issues outlined in the previous section, we have developed a system for high-level decision-making capabilities for future Mars rovers. The system framework is shown in Figure 1, and currently is comprised of three major components:

- **A Continuous Planner** that provides capabilities for initial rover plan generation and the continuous modification of that plan based on changing operating context and goal information.
- **A Reactive Task-Level Executive** that provides task-level control capabilities, including execution and monitoring of a rover plan, as well as providing mediation between a planner and low-level robot functionality.
- **A Global Path Planner** that provides path planning information about predicted routes to both the planner and executive.

We begin our system description by first introducing the underlying robotic architecture used to support this work. Next we describe our framework for integrating planning and execution capabilities. We then give a more detailed description of the individual system modules.

### CLARAty Architecture

The Coupled Layered Architecture for Robotic Autonomy (CLARAty) (Volpe, et al., 2001) is being developed at JPL in response to the need for a robotic control architecture that can support future mission autonomy requirements. CLARAty uses a two-layered approach to organizing robotic capabilities, which is an evolution of the traditional three-layer architecture. The top Decision Layer (Estlin, et al., 2001) contains techniques for autonomously creating and carrying out sequences of rover actions that will achieve an input set of goals. It also provides a framework for using different types of planning and executive systems, and for enabling new ways of combining such systems. The bottom Functional Layer (Nesnas et al., 2001) provides a set of standard, generic robot capabilities that interface to system hardware.

Though CLARAty is a two-layered architecture it has been designed to support both traditional three-layer approaches to robotic control (which have a planner, executive and control layer) (Alami, et al., 1998; Bonasso, et al., 1998; Jonsson et al., 2000) as well as support new research in the field that is closely integrating the planning and execution processes (Myers, 1998; Chien et al., 2000; Fisher et al., 2002) and has moved towards collapsing the planning and executive layers into one.

In this paper we are focusing on the first instantiation of the CLARAty Decision Layer, which is provided by the CLEaR unified planning and execution framework. Since CLEaR is integrated as part of the CLARAty architecture, it uses the CLARAty Functional Layer to both command the rover and access information about rover state including real-time updates and feedback.

### CLEaR Framework

The CLEaR (Closed-Loop Execution and Recovery) unified planning and execution framework (Fisher, et al., 2002) was developed to pursue a tight integration of planning and execution capabilities. Currently, CLEaR is a hybrid controller system that is built on top of the CASPER (Continuous Activity Scheduling, Planning, Execution and Re-planning) continuous planner and the TDL (Task Description Language) executive system, which are described further below. Previous versions of the CLEaR framework have been demonstrated for Deep Space Network (DSN) antenna control (Fisher, et al., 2000). Currently CLEaR is being extended to provide planning and execution support for planetary rovers.

CLEaR's primary objective is to provide a tightly coupled approach to coordinating goal-driven and event-driven behavior. Many past approaches have followed a three-level architecture style where the planning and executive processes are treated as *black box* systems. This is in contrast to how CLEaR enables the planner and executive to interact with each other and more effectively share the responsibility for decision making. In part this is managed through shared plan information and continual updates of state being made available to both the planner and executive. CLEaR also provides heuristic support for deciding when certain plan conflicts should be handled by the planner vs. the executive. For instance if a rover gets off track during a traverse, the reaction of the planner and executive need to be coordinated. If the executive believes it can resolve the navigation delay within the planned time constraints it will manage the plan changes. However, once the executive identifies that the repair will require more time or resources than allotted by the planner, it will then fail the task, which will result in the planner using its global perspective to fix the problem.

Planning in CLEaR is provided by the CASPER continuous planning system (Chien, et al., 2000). Based on an input set of science goals and the rover's current state, CASPER generates a sequence of activities that satisfies the goals while obeying relevant resource, state and temporal constraints, as well as operation (or flight) rules. Plans are produced using an *iterative repair* algorithm that classifies conflicts and resolves them individually by performing one or more plan modifications. CASPER also monitors current rover state and the execution status of plan activities. As this information is acquired, CASPER updates future-plan projections. Based on this new information, new conflicts and/or opportunities may arise, requiring the planner to re-plan in order to accommodate the unexpected events.

The executive functionality in CLEaR is performed by the TDL executive system (Simmons and Apfelbaum, 1998). TDL was designed to perform task-level control for a robotic system and to mediate between a planning system and low-level robot control software. It expands abstract tasks into low-level commands, executes the commands, and monitors their execution. It also provides direct support for exception handling and fine-grained synchronization of subtasks. TDL is implemented as an extension of C++ that simplifies the development of robot control programs by including explicit syntactic support for task-level control capabilities. It uses a construct called a *task tree* to describe the tree structure that is produced when tasks are broken down into low-level commands.

For the work described in this paper, we used an early version of CLEaR, which integrated a planner and executive (i.e. CASPER and TDL) as separate modules. In this version, our approach is similar to that of previous three-layer architectural approaches. However, as compared to other three-layer approaches where planning is typically done in a batch fashion and takes on the order of minutes to hours, this integration uses a continuous planning approach, where plans are updated and repaired in a matter of seconds. This enables CLEaR to use planning techniques at a finer timescale for tracking the progress of plan execution, quickly identifying potential problems in future parts of the plan, and responding accordingly. As we expect minor portions of the plan to change frequently, we use a lightweight plan runner to dispatch activities to the executive a few seconds before the task's scheduled *start time*. This approach differs from the more common batch approach of turning the entire plan over to the executive for execution. Executive techniques are then used in only reactive situations or at times where procedural reasoning is preferred. In the Discussion section, we discuss other steps we have taken towards tighter integration that have been tested in simulation.

Another way that CLEaR differs from previous approaches is in how the delegation between the planner and the executive is managed. We have primarily taken a planning centric approach to this management. The planner handles the decision of when an activity should be mapped to a task for execution as well as when to perform re-planning. The re-planning process is driven by applying and propagating updates to the plan, and then taking corrective actions to address any conflicts or opportunities that may arise. Re-planning can also be performed synchronously with any already executing tasks. Once a planning activity has been mapped to an executive task for execution, control over that one task is given to the executive. The executive may then perform further task expansions as a result of updates and/or exception handling. The executive also provides task completion status back to the planner by either marking an activity as complete or failed. A task is marked as completed when the executive decides the task has met its objective, or marked as failed upon concluding that given constraints provided by the planner cannot (or even might not) be met.

## Global Path Planning

To provide spatial reasoning capabilities to the CLEaR system, we are also employing a global path-planning module, which provides rover route information to the planner and executive based on a map of the rover's environment. This module is intended to give a global perspective of the rover's anticipated path as opposed to the local perspective that would be considered by obstacle avoidance software. We are assuming that for most rover operations some global map information would be available through orbital or descent imagery, or from panoramic imagery generated onboard the rover itself. We are also assuming that much of the global map information would be at a low resolution and thus a significant number of terrain features or obstacles may be missing and will need to be considered dynamically.

Currently, CASPER and TDL query for two main pieces of information from the path-planning module. The first type of information is estimated distances between science targets and other major waypoints. The second type is a list of intermediate-waypoint coordinates that can be used to direct the rover's traverse to a particular target. Path-distance information is used by the planner to estimate the duration and power required for rover traverses between targets. Intermediate waypoints are used by the executive to track the rover's progress during a traverse. During a traverse, the executive monitors the progress of the rover as position updates come in. Because the executive knows the nominal velocity of the rover and the distance it must cover, it can predict how long it will take to reach its goal. If the executive anticipates that the traverse will take longer than the allotted time from the planner, then the executive may request new waypoints or it may halt the traverse and trigger the creation of a new plan.

For the tests reported in this paper, we used an implementation of the Tangent Graph path-planning algorithm (Latombe, 1991) to provide global path-planning capabilities. Tangent Graph operates by building a path through map free space as represented in a reduced visibility graph of 2-D polygonal obstacles. We are also currently extending our path-planning module to use other type of path planners.

## Rover Scenario Testing

To test and validate our approach to planning and execution for rover operations, we are developing a number of rover scenarios that attempt to emulate mission conditions and goals. This section describes the results of testing with one particular scenario using two different rovers. We have also tested our system using the ROAMS rover simulation tool (Yen & Jain, 1999), however we only focus on our hardware testing experiences for this paper.

### Scenario Description

Figure 2A shows a map of the testing scenario. A number of science targets are identified on the map and dark shapes represent obstacles known *a priori* (e.g., from descent or orbital imagery). This map represents a sample mission-site location that would be explored in detail where data would be gathered using multiple instruments at a number of locations.

In this scenario, the types of science performed at the site include images (taken with a mast camera), spectrometer reads, and digs. An end of day communication activity with Earth is also required (though not reflected on the initial scenario map), and must be scheduled in a certain time window. Communication activities typically require a significant amount of power, thus the inclusion of this activity affects what science operations can be performed. We also made several assumptions in developing this scenario. One, we assume mission scientists have assigned a priority to each science target. Two, we assume that some map information is known ahead of time although other obstacles likely exist that are not identified on the original map and thus, need to be detected through rover sensors and obstacle avoidance software. Three, more science goals are specified than can be achieved given the resource levels allotted for the scenario time period and the rover planning software is responsible for deciding what subset of those goals will be achieved.

The model developed for the planner contains 15 different types of activities where 6 of those are executable activities (e.g., science activities, navigation activities) and the rest are used to model exogenous processes or events (such as when the communication window with Earth opens and closes). A number of operation and resource constraints for these activities are also represented and maintained by the planner. Examples of constraints include ensuring limited memory and energy resources are not oversubscribed, ensuring that the rover is in the correct position and orientation for science operations, and handling a limited communication window with Earth. When executed, planning activities are mapped to tasks in the executive. The model developed for the executive contains 9 different task definitions and 10 monitor definitions, where a monitor handles the tracking and evaluation of state data from the low-level software to detect events such as when a task has successfully completed or when an error has occurred. Both planning and executive systems also monitored several rover states and resources, including current position, heading, energy level and memory level.

### Testing Environment and Setup

To evaluate our system, we performed a series of tests in the JPL Mars Yard using two different rovers, Rocky 7 and Rocky 8, which are shown in Figure 3. Rocky 7 is approximately the same size and mass as the Mars Pathfinder rover, Sojourner. It employs a rocker-bogie six-wheel configuration, and is a partially-steered vehicle, where it only has steering capability on two corners. In contrast, Rocky 8 is roughly an order of magnitude larger than Rocky 7 and is similar in size to the twin MER rovers. Rocky 8 also employs a rocker-bogie six-wheel configuration, however it is a fully-steered vehicle with all-wheel drive and all-wheel steering. During testing, the planning and executive systems ran on an offboard workstation that communicated with the rovers using Wireless Ethernet. The CLEaR system runs on both Sun

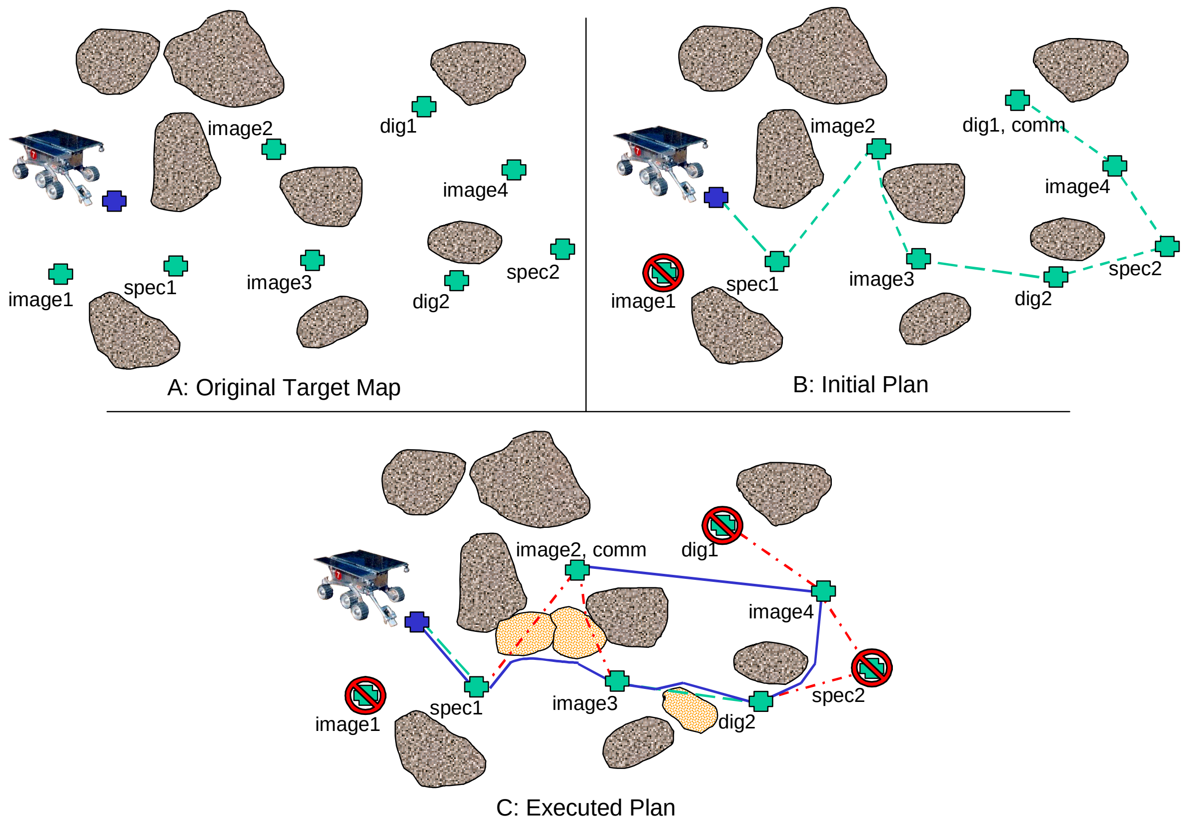

**Figure 2: Sample Scenario Map for a Geological Site Exploration**

Solaris and Linux operating systems. Performance numbers reported in this paper were run on a Linux 1.7 GHz Pentium 4 workstation. The CLARAty low-level control software was run onboard each rover.

During plan generation and re-planning, science-target visits are ordered by the planner's TSP (Traveling Salesman Problem) heuristic solver so that the rover is choosing the shortest path allowable by constraints and based on its current map information. Currently, the path-planner module is used to provide distance estimates for TSP. Though this requires the system to call the path planner on each pair of goals, the overall time required is small due to the relative speed of the path planner (i.e., each distance query took around .03 secs to complete). Furthermore, for a science site investigation, it is typical to only have a relatively small number of goals in the immediate area that are being considered. Thus, this testing simulates a realistic number of calls to the path planner.

We used two different modes of navigation on the rovers. For Rocky 7, obstacle avoidance capabilities were not available so we simulated a simple obstacle-avoidance behavior. If an obstacle appeared in the rover path and was in fairly close-range, we would manually abort the current move command and update the global map to allow the path planner to select a path around the obstruction. For Rocky 8, we used the GESTALT navigation system (Goldberg et al., 2002), which is providing obstacle avoidance and navigation capabilities to the MER rovers. Based on local terrain knowledge, the obstacle avoidance software decides the best direction for the rover to move that will allow the rover to efficiently reach its goal waypoint while avoiding obstacles or hazardous terrain.

Several other activities not available at that time were simulated for testing. First, the global map of the scenario was manually created based on the scenario rock layout in the Mars Yard. Second, science and communication operations were simulated since these modules are not currently available through the CLARAty Functional Layer. At each science target, the rover would stop for an appropriate amount of time, and energy and memory use for each science operation was estimated and reflected in state updates.

**Figure 3: Rocky 7 and Rocky 8 Rovers**

## Testing Results

Figures 2B and 2C show the results from running Rocky 8 on this testing scenario. Figure 2B shows the results of initial plan generation. The plan contained 53 different activity instances and took the planner 3.7 seconds to construct. As previously mentioned, more science goals have been provided to the planner than can be supported by onboard memory and energy levels. Thus, CASPER excludes a low priority science target from the initial plan due to an energy conflict in order to allow enough energy to complete the remaining science activities as well as the end-of-day communication activity, which is performed at the same location as the last science target.

Figure 2C shows the results of plan execution and re-planning. The solid line shows the actual path of Rocky 8 during scenario execution. The dashed lines show what the rover's planned path was at different stages of execution. There are several points of execution where either the CASPER planner or the TDL executive revised the rover's plan based on current state and resource information. Each re-plan took an average of 2 seconds to complete.

The first point of plan revision is during the traverse between the *spec1* and *image2* targets. During this traverse the rover encounters several unexpected obstacles that block its path to the next target. The navigation system causes the rover to veer off its planned path and attempts to find a new route to the goal location. TDL monitors this path change and iteratively checks how far behind schedule the rover has fallen. Once TDL estimates the rover will not be able to complete the current move activity within an allowable time range, it halts the current move and signals to CASPER that the move activity has failed. CASPER then repairs the plan, taking new map information about the obstacles into account, and finds that a new target ordering will still achieve all remaining science targets.

The second point of plan revision comes after the completion of the *image3* science activity. For each science activity, an expected duration and resource usage has been encoded in the planner's model of rover operations. However, since these values cannot always be accurately predicted, they are monitored during execution and the planner stands ready to update the plan based on new information. At the *image3* science activity, the Functional Layer simulates that the acquired image data cannot be compressed as much as originally estimated. The new memory level is forwarded from the Functional Layer to TDL and then to CASPER, which updates the activity plan. This update causes a plan conflict to arise since now memory will be oversubscribed before the plan is completed. CASPER resolves this conflict by deleting a later spectrometer read (*spec2*), which ensures enough memory is available to collect data at the remaining targets. Again, CASPER deletes a low-priority science activity and attempts to preserve as many high-priority activities as possible. After the science activity is deleted, the plan is updated to reflect new traverse routes between science targets.

A third revision occurs during the traverse from *image3* to *dig2*. Another unexpected obstacle is encountered and obstacle avoidance software moves the rover off the planned path. TDL monitors rover progress and finds that this time there is enough time to avoid the obstruction and no re-planning on CASPER's part is required.

The last point of plan revision comes after the completion of the *dig1* science activity. This situation is similar to the previously explained memory over-subscription, however this time the activity uses more energy than anticipated causing a conflict. Again, CASPER resolves this conflict by deleting one of the remaining lower-priority science goals whose deletion will release enough energy to successfully complete the communication activity. The *dig2* science activity is deleted and a new path is calculated to the remaining targets. The remainder of the plan then executes as expected.

## Discussion

Though testing of this scenario was successful for both rovers, we did encounter a number of issues that need to be resolved in order to provide a more robust and stable system. Some of these issues are particular to our approach, however, many of them will apply to the general use of planning and execution techniques to this application area.

One issue that consistently arose was the planning and execution system's reliance on accurate position estimation. This reliance affects not only the estimated durations and power requirements for a traverse but also affects the system's determination of whether an activity has completed successfully. There are several factors contributing to this issue. One factor was that only very limited position estimation capabilities were available on Rocky 7 and 8 during this testing. Position estimation was based solely on wheel odometry, which can incur significant drift error, especially when navigating on sand or over rocks. Currently, CLARAty is developing a more sophisticated position estimation approach that includes a Kalman-Filter technique and the use of additional sensor data.

Although we can expect some improvements from the estimation software, the issue remains that planning and execution software cannot expect perfect position estimates and this software must be flexible enough to operate using uncertain state information. For these tests, we added flexibility to our plan in two simple ways. One, we adjusted the determination of a successful traverse so that the rover was only required to be in a certain neighborhood of a goal target or waypoint for that traverse to be considered successful. Two, we had the planning system automatically build in some buffer room between plan activities so that slightly longer-than expected traverses did not disrupt the overall plan. Development of a more robust and efficient solution to the problem of shifting a grounded plan (due to overruns or underruns) is an area of future work. One approach taken in related planning work is to generate and execute a temporally flexible plan (Jonsson, et al., 2000). However, working with a temporally grounded plan has been shown to enable fast planning and re-planning speeds and thus contributes to the responsiveness of the system. We are currently investigating a hybrid approach to this problem where plans are generated and repaired using a grounded-time representation, and are mapped to flexible time for execution using the underlying plan constraints.

A second identified issue was that the executive could only be given limited ability to modify the plan since it had little or no knowledge of many state and resource constraints maintained by the planner. Currently, the domain knowledge encoder must ensure that the executive can only modify the plan within certain limits to ensure that no operations constraints are violated. In our scenario testing, we added some simple heuristic knowledge to the executive that defined when it could attempt local fixes and when it needed to fail an activity and ask the planner to replan.

We have further addressed this issue in recent work by developing two techniques that enable the executive to take a larger role in plan modification: 1) Atomic Resource Manager (ARM) and 2) Execution Time Query (ETQ) (Fisher, et al., 2002). Both techniques have been tested in simulation and in future work we will experiment using rover hardware.

ETQ enables the executive to use the planner's constraint knowledge to "ask permission'' to violate planner-imposed constraints on a task. This gives the executive the ability to successfully complete a task that is using an unexpected amount of a resource. If some look ahead is encoded into the executing tasks then this query can be made in advance of the constraint violation. Through the use of the query mechanism this allows the unified system to perform an appropriate action. This action might be to abandon the current task or might be to adjust the remainder of the plan to accommodate the current task with its modified constraints. If the system chooses to abandon the current task, then CASPER will produce a newly modified sequence and provide the executive with the new course of action.

ARM provides additional capabilities to the executive by allowing it to coordinate tasks that require intermittent use of some resource. We have developed a rapid scheduling algorithm for managing the intermittent allocation of an atomic resource (such as a camera) to more than one concurrently running task.

## Future Work

In the previous section, we identified issues that arose during testing and discussed progress that we have made to address them. We have also identified several areas of future work.

One of the ongoing goals of the CLEaR system is to tightly unify the planning and execution processes. A step towards this goal is to make the executive aware of constraints represented or generated by the planner. The ARM and ETQ techniques presented in the Discussion section represent further progress toward our objective. Also previously mentioned is the ongoing work in the area of grounded time planning and flexible time execution.

Another step towards integrating planning and executive is to enable procedural capabilities to be accessed by the planner during plan generation and repair. This step will enable procedural constructs, such as conditionals, to be easily used during plan search. These types of constructs are difficult to represent in a declarative representation, however as previously mentioned, it would be beneficial to have such constructs when reasoning about certain activity types at the planning level. Eventually, we hope to fully integrate CASPER and TDL, where both planning and executive functionality use a shared representation and operate on one planning database. This integration would alleviate the need for two different domain models and would enable planning and executive capabilities to more easily interact and operate on all levels of activity granularity.

We also plan to develop more sophisticated techniques for dealing with activity/task failures involving exception handling. For instance, our system should handle science operations failing in different fashions such as an unsuccessful science data acquisition (e.g., an over-exposed or miss-targeted frame or an unsuccessful grasping of a rock). While we already have the mechanisms in place to handle retry-type recoveries, such would be used in the rock example, however procedures for other types of exception handling will likely require extensions to our current system. For instance, the planning and execution system may need to closely coordinate with other onboard software that can evaluate whether a science operation was successful.

Another area of current work is to provide more realistic map information to the planning and execution system and to incorporate different types of path planners. For instance, we plan to interact with the TEMPEST path planner (Tompkins, et al., 2001), which can input additional constraints that affect the path selection process. Most path planners search for only the shortest path between two points, however there are many other constraints that may affect path selection for rovers, such as shadowing, communication opportunities, terrain risk, etc. Constraints that are important for the current rover plan could be identified by the activity planner and used to focus the path search in TEMPEST.

One last area of future work is to dynamically identify times that new science goals could be added to the plan. In our past scenario, we focused on how to repair the plan when things went wrong, which usually resulted in low-priority science activities being discarded from the plan. We would also like our planning and execution system to be able to handle situations were things go better than expected. For example, a rover traverse may be much shorter than expected or a new and high-priority science opportunity could dynamically arise that wasn't previously identified. In both of these situations the planning and execution system could improve the quality of the plan by adding in additional science activities. These could be previously requested science observations that were discarded due to limited resource availability, or brand new opportunities that rover sensors or onboard data analysis algorithms have identified as valuable.

## Related Work

A number of planning and executive systems have been successfully used for robotic applications and have similarities to the approach we describe in this paper. Most of these approaches have used some combination of planning and execution, however they differ in not only the behavior of these individual components, but also in how these systems interface with each other and with other system modules.

The Remote Agent Experiment (RAX) (Jonsson, et al., 2000) was flown on the NASA Deep Space One (DS1) mission. It demonstrated the ability of an AI system to respond to high-level spacecraft goals by generating and executing plans onboard the spacecraft. The planner in RAX takes as input a schedule request and produces a flexible, temporal schedule for execution by its executive. A major limitation to this approach was that planning was only performed in a batch fashion. If re-planning was required, the spacecraft was "safed" until a new plan had been generated (which could be on the order of hours). Furthermore, since RAX was applied to a spacecraft, it did not handle issues with surface navigation and path planning.

Another approach directed towards rover command generation uses a Contingent Planner/Scheduler (CPS) that was developed to schedule rover-scientific operations using a Contingent Rover Language (CRL) (Bresina, et al, 1999). CRL allows both temporal flexibility and contingency branches in rover command sequences. Contingent sequences are produced by the CPS planner and then are interpreted by an executive, which executes the final plan by choosing sequence branches based on current rover conditions. In this approach, only the executive is onboard the rover; planning is intended to be a ground-based operation. Since only a limited number of contingencies can be anticipated, our approach provides more onboard flexibility to new situations. In the CRL approach, if a situation occurs onboard for which there is not a pre-planned contingency, the rover must be halted to wait for communication with ground.

Other similar approaches include Atlantis (Gat, 1998), 3T (Bonasso, et al., 1997), and a robotic control architecture developed at the LAAS-CNRS lab (Alami, et al., 1998) which all use a deliberative planner and executive (or sequencing component) on top of a set of reactive controllers. These approaches have distinctly separated planning and execution techniques, and have not closely interacted with navigation software used for rover missions. Also, the CPEF (Continuous Planning and Execution Framework) (Myers, 1998) is a similar framework to CLEaR for combining planning and execution. However, CPEF is designed to cull out key aspects of the world to monitor and has been primarily tested in military air-campaign domains.

## Conclusions

This paper discusses a number of challenges for using planning and execution techniques to provide autonomous rover capabilities for future NASA missions. We describe our approach for using an onboard planning and execution system and explain how it provides capabilities for sequence generation, execution, monitoring, and re-planning. We also describe how our system interacts with other software modules such as path planning and low-level control software. Finally we discuss our experiences with testing our planning and execution system in providing decision-making capabilities for two JPL rovers.

## Acknowledgements

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

## References

Alami, R., Chautila, R., Fleury, S., Ghallab, M., and Ingrand, F., “An Architecture for Autonomy,” *International Journal of Robotics Research*, 17(4) April, 1998.

Bonasso, R., Firby, R., Gat, E., Kortenkamp, D., Miller, D., and Slack, M., “Experiences with an Architecture for Intelligent, Reactive Agents,” *Journal of Experimental and Theoretical Artificial Intelligence Research*, 9(1), 1997.

Bresina, J., Golden, K., Smith, D., and Washington, R., “Increased Flexibility and Robustness of Mars Rovers,” *Proceedings of the International Symposium, on AI, Robotics and Automation for Space*, Noordwijk, The Netherlands, June 1999.

Chien, S., Knight, R., Stechert, S., Sherwood, R., and Rabideau, G., "Using Iterative Repair to Improve Responsiveness of Planning and Scheduling," *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling*, Breck-enridge, CO, April 2000.

Estlin, T., Volpe, R., Nesnas, I., Mutz, D., Fisher, F., Engelhardt, B., and Chien, S. “Decision-Making in a Robotic Architecture for Autonomy,” *Proceedings of the International Symposium, on AI, Robotics and Automation for Space*, Montreal, Canada, June 2001.

Fisher, F., Knight, R., Engelhardt, B., Chien, S., and Alejandre, N., “A Planning Approach to Monitor and Control for Deep Space Communications,” *Proceedings of the 2000 IEEE Aerospace Conference*, Big Sky, Montana, March 2000.

F. Fisher, D. Gaines, T. Estlin, S. Schaffer, C. Chauinard, "CLEaR: A Framework for Balancing Deliberation and Reactive Control," *Proceedings of the AIPS On-line Planning and Scheduling Workshop*, Toulouse, France, April 2002

Gat, E.., “ESL: A Language for Supporting Robust Plan Execution in Embedded Autonomous Agents,” *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, July 1992.

Goldberg, S., Maimone, M., and Matthies, L. “Stereo Vision and Rover Navigation Software for Planetary Exploration,” *Proceedings of the 2002 IEEE Aerospace Conferenc*e, Big Sky, Montana, Match, 2002.

Jonsson, A., Morris, P., Muscettola, N., Rajan, K., and Smith, B., "Planning in Interplanetary Space: Theory and Practice," *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems, Breckenridge*, CO, April 2000.

Latombe, J., *Robot Motion Planning*, New York, Kluwer, 1991.

Mishkin, A., Morrison, J., Nguyen, T., Stone, H., Cooper, B., Wilcox, B., "Experiences with Operations and Autonomy of the Mars Pathfinder Microrover," *Proceedings of the 1998 IEEE Aerospace Conference*, Aspen, CO, March 1998.

Nesnas, I., Volpe, R., Estlin, T., Das, H., Petras, R., and Mutz, D., “Toward Developing Reusable Software Components for Robotic Applications,” *Proceedings of the International Conference on Intelligent Robots and Systems*,Maui, Hawaii, Nov 2001.

Myers, K. “Towards a framework for continuous planning and execution.” *Proceedings of the AAAI 1998 Fall Symposium on Distributed Continual Planning*, Menlo Park, CA, 1998.

Simmons, R. and Apfelbaum, D., “A Task Description Language for Robot Control,” *Proceedings of the Intelligent Robots and Systems Conference*, Vancouver, CA, October 1998.

Tompkins, P, Stentz, A., and Whitaker, W., “Automated Surface Mission Planning Considering Terrain, Shadows, Resource and Time,” *Proceedings of the International Symposium on AI, Robotics and Automation in Space*, Montreal, Canada, June 2001.

Volpe, R., Nesnas, I., Estlin, T., Mutz, D., Petras, R., Das, H., "The CLARAty Architecture for Robotic Autonomy," *Proceedings of the 2001 IEEE Aerospace Conference*, Big Sky, Montana, March 2001.

Yen, J., and Jain, A., “ROAMS: Rover Analysis Modeling and Simulation Software,” *Proceedings of the International Symposium on AI, Robotics and Automation in Space*, The Netherlands, June 1999.