AN AUTOMATED ROVER COMMAND GENERATION PROTOTYPE FOR THE MARS 2003 MARIE CURIE ROVER

Rob Sherwood, Andrew Mishkin, Tara Estlin, Steve Chien,
Scott Maxwell, Barbara Engelhardt, Brian Cooper, Gregg Rabideau

Jet Propulsion Laboratory
4800 Oak Grove Dr.
Pasadena, CA 91109
818-393-5378
firstname.lastname@jpl.nasa.gov

Abstract

This paper discusses a proof-of-concept prototype for ground-based automatic generation of validated rover command sequences from high-level science and engineering activities. This prototype is based on ASPEN, the Automated Scheduling and Planning Environment. This Artificial Intelligence (AI) based planning and scheduling system will automatically generate a command sequence that will execute within resource constraints and satisfy flight rules. Commanding the rover to achieve mission goals requires significant knowledge of the rover design, access to the low-level rover command set, and an understanding of the performance metrics rating the desirability of alternative sequences. It also requires coordination with external events such as orbiter passes and day/night cycles. An automated planning and scheduling system encodes this knowledge and uses search and reasoning techniques to automatically generate low-level command sequences while respecting rover operability constraints, science and engineering preferences, and also adhering to hard temporal constraints. Enabling goal-driven commanding of planetary rovers by engineering and science personnel greatly reduces the requirements for highly skilled rover engineering personnel and Rover Science Team time. This in turn greatly reduces mission operations costs. In addition, goal-driven commanding permits a faster response to changes in rover state (e.g., faults) or science discoveries by removing the time consuming manual sequence validation process, allowing rapid "what-if" analyses, and thus reducing overall cycle times.

Introduction

Unlike more traditional deep space missions, surface roving missions must be operated in a reactive mode, with mission planners waiting for an end of day telemetry downlink--including critical image data--in order to plan the next day’s worth of activities. Communication time delays over interplanetary distances preclude simple 'joysticking' of the rover. A consequence of this approach to operations is that the full cycle of telemetry receipt, science and engineering analysis, science plan generation, command sequence generation and validation, and uplink of the sequence, must typically be performed in twelve hours or less. Yet current rover sequence generation is manual (Mishkin, et al., 1998), with limited ability to automatically generate valid rover activity sequences from more general activities/goals input by science and engineering team members. Tools such as the Rover Control Workstation (RCW) and the Web Interface for Telescience (WITS) provide mechanisms for human operators to manually generate plans and command sequences. (Backes, et. al, 1998) These tools even estimate some types of resource usage and identify certain flight rule violations. However, they do not provide any means to modify the plan in response to the constraints imposed by available resources or flight rules, except by continued manual editing of sequences. This current situation has two drawbacks. First, the operator-intensive construction and validation of sequences puts a tremendous workload on the rover engineering team. The manual process is error-prone, and can lead to operator fatigue over the many months of mission operations. Second, the hours that must be reserved for sequence generation and validation reduces the time available to the science team to identify science targets and formulate a plan for submission to the engineering team. This results in reduced science return. An automated planning tool would allow the science team and sequence team to work together to optimize the plan. Many different plan options could be explored. The faster turnaround of automated planning also permits shorter than once a day planning cycles.

The Rover Control Workstation (RCW) tool, used to operate the Sojourner rover during the Pathfinder mission, provides visualization for vehicle traverse (movement) planning, a command interface, constraint checking for individual commands, and some resource estimation (for sequence execution time and telemetry volume). However, this tool was never intended for automated goal-based planning of rover activities. To deal with these issues, there is a need for a new tool that is specifically geared toward automated planning.

We are using AI planning/scheduling technology to automatically generate valid rover command sequences from activity sequences specified by the mission science and engineering team. This system will automatically generate a command sequence that will execute within resource constraints and satisfy flight rules. Commanding the rover to achieve mission goals requires significant knowledge of the rover design, access to the low-level rover command set, and an understanding of the performance metrics rating the desirability of alternative sequences. It also requires coordination with external events such as orbiter passes and day/night cycles. An automated planning and scheduling system encodes this knowledge and uses search and reasoning techniques to automatically generate low-level command sequences while respecting rover operability constraints, science and engineering preferences, and also adhering to hard temporal constraints. A ground-based interactive planner combines the power of automated reasoning and conflict resolution techniques with the insights of the Science Team or Principal Investigator (PI) to prioritize and re-prioritize mission goals.

ASPEN Planning System

Planning and scheduling technology offers considerable promise in automating rover operations. Planning and scheduling rover operations involves generating a sequence of low-level commands from a set of high-level science and engineering goals.

ASPEN (Chien, et al., 2000; Fukanaga, et al., 1997; Rabideau, et al., 1999) is an object-oriented planning and scheduling system that provides a reusable set of software components that can be tailored to specific domains. These components include:

- An expressive constraint modeling language to allow the user to define naturally the application domain
- A constraint management system for representing and maintaining spacecraft and rover operability and resource constraints, as well as activity requirements
- A set of search strategies for plan generation and repair to satisfy hard constraints
- A language for representing plan preferences and optimizing these preferences
- A soft, real-time replanning capability
- A temporal reasoning system for expressing and maintaining temporal constraints
- A graphical interface for visualizing plans/schedules (for use in mixed-initiative systems in which the problem solving process is interactive).

In ASPEN, the main algorithm for automated planning and scheduling is based on a technique called *iterative repair* (Zweben et al., 1994). During iterative repair, the conflicts in the schedule are detected and addressed one at a time until conflicts no longer exist, or a user-defined time limit has been exceeded. A conflict is a violation of a resource limitation, parameter dependency or temporal constraint. Conflicts can be repaired by means of several predefined methods. The repair methods are: moving an activity, adding a new instance of an activity, deleting an activity, detailing an activity, abstracting an activity, making a resource reservation of an activity, canceling a reservation, connecting a temporal constraint, disconnecting a constraint, and changing a parameter value. The repair algorithm may use any of these methods in an attempt to resolve a conflict. How the algorithm performs is largely dependent on the type of conflict being resolved.

Rover knowledge is encoded in ASPEN under seven core model classes: activities, parameters, parameter dependencies, temporal constraints, reservations, resources and state variables. An activity is an occurrence over a time interval that in some way affects the rover. It can represent anything from a high-level goal or request to a low-level event or command. Activities are the central structures in ASPEN, and also the most complicated. Together, these constructs can be used to define rover procedures, rules and constraints in order to allow manual or automatic generation of valid sequences of activities, also called plans or schedules.

Once the types of activities are defined, specific instances can be created from the types. Multiple activity instances created from the same type might have different parameter values, including the start time. Many camera-imaging activities, for example, can be created from the same type but with different image targets and at different start times. The sequence of activity instances is what defines the plan.

The flight rules and constraints are defined within the activities. The flight rules can be defined as temporal constraints, resource constraints, or system state constraints. Temporal constraints are defined between activities. An example would be that the rate sensor must warm up for two to three minutes before a rover traverse. In ASPEN, this would be modeled within the "move rover" activity as shown in Figure 1. The rate_sensor_heat_up is another activity that is presumed to turn on a rate sensor heater.

Constraints can also be state or resource related. State constraints can either require a particular state or change to a particular state. Resource constraints can use a particular amount of a resource. Resources with a capacity of one are called atomic resources. ASPEN also uses non-depletable and depletable resources. Non-depletable resources are resources that can used by more than one activity at a time and do not need to be replenished. Each activity can use a different quantity of the resource. An example would be the rover solar array power. Depletable resources are similar to non-depletable except that their capacity is diminished after use. In some cases their capacity can be replenished (memory capacity) and in other cases it cannot (battery energy, i.e. non-rechargeable primary batteries). Resource and state constraints are defined within activities using the keyword "reservations." See Figure 1 for an example.

```
Activity move_rover {
  constraints =
    starts after end_of rate_sensor_heat_up by [2m,3m];
  reservations =
    solar_array_power use 35,
    rate_sensor_state change_to "on",
    target_state must_be "ready";
};
```

Figure 1 - ASPEN Modeling Language Example

The job of a planner/scheduler, whether manual or automated, is to accept high-level goals and generate a set of low-level activities that satisfy the goals and do not violate any of the rover flight rules or constraints. ASPEN provides a Graphical User Interface (GUI) for manual generation and/or manipulation of activity sequences. Figure 2 contains a screen dump of the GUI.

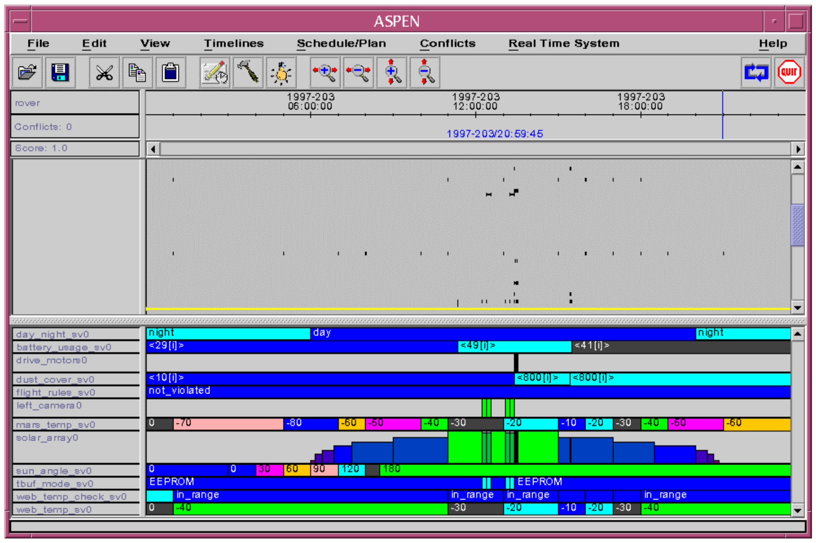

Figure 2 - ASPEN GUI

Mars Surveyor Lander

The Mars Surveyor 2001 Lander was scheduled for launch in April 2001. Due to a reorganization of the Mars Exploration Program at JPL, this launch was cancelled. The mission design for the next launch opportunity, in 2003, is currently being reviewed. One option under consideration is to launch a modified version of the 2001 Lander with a payload complement including the Marie Curie rover. For the purposes of this paper, we are assuming the 2001 Lander configuration with a 2003 arrival date. If the lander design changes, we will update our planner models accordingly.

The lander will carry an imager to take pictures of the surrounding terrain during its rocket-assisted descent to the surface. The descent-imaging camera will provide images of the landing site for geologic analyses, and will aid planning for initial operations and traverses by the rover. The lander will also be a platform for instruments and technology experiments designed to provide key insights to decisions regarding successful and cost-effective human missions to Mars. Hardware on the lander will be used for an in-situ demonstration test of rocket propellant production using gases in the Martian atmosphere. Other equipment will characterize the Martian soil properties and surface radiation environment. Figure 3 contains a diagram of the lander and instruments. The Marie Curie rover will be deployed using a robotic-arm attached to the lander.

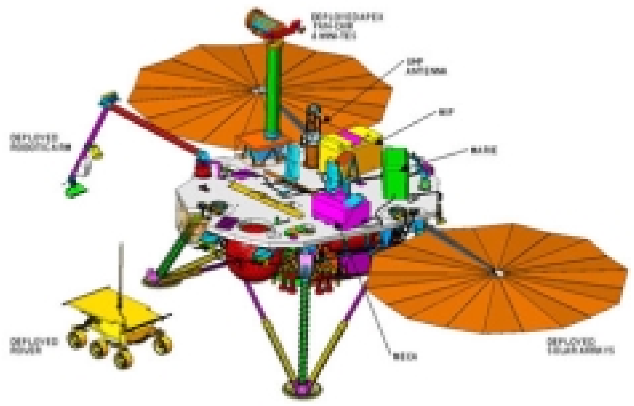

Figure 3 - Mars 2001 Lander

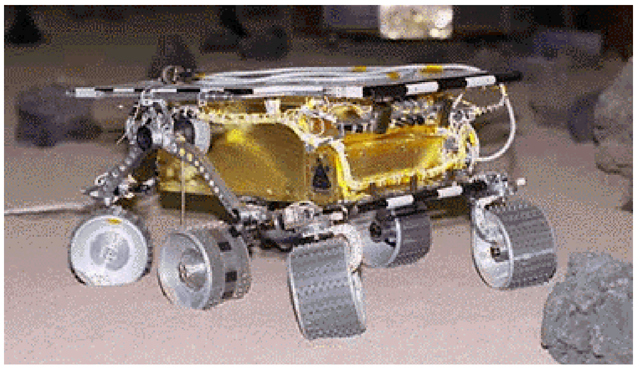

Figure 4 - Marie Curie Rover

The Marie Curie rover is very similar to the Mars Pathfinder Sojourner rover. (See Figure 4.) In fact, it is the same rover that was used in the Pathfinder test bed during the mission. (Mishkin et al., 1998; Mishkin 1998) Additional modifications have been made to accommodate the robotic-arm-based deployment from the 2001 Lander. In addition, some minor engineering enhancements have been added. A description of the rover components is included in Table 1.

- 6-Wheeled robotic vehicle, rocker-bogie mobility chassis
- Mass: 10.5 kilograms
- Deployed volume: 65cm (l) by 48cm (w) by 30cm (h).
- Intel 80C85 CPU (~100Kips), 16K PROM, 64K rad hard RAM, 176K EEPROM, 512K RAM
- Forward Black & White stereo cameras, and rear B&W mono camera
- GaAs solar panel (16W peak)
- Primary (non-rechargeable) batteries
- UHF Radio Modem
- Laser stripers for hazard detection

Table 1 - Rover Description

Model Description

The Marie Curie planning model was built to a level at which all flight rules and constraints could be implemented. The resources include the three cameras, Alpha Proton X-Ray Spectrometer (APXS), APXS deploy motor, drive motors, solar array, battery, RAM usage, and EEPROM usage.

There are 27 different state variables used to track the status of various devices, modes, and parameters. Some of these parameters map directly onto rover internal parameters and others are related to the ASPEN specific model. We are not modeling all rover internal parameters because many are not useful for automating planning. We have defined 162 activities of which 63 decompose directly into low-level rover commands.

There are several constraints that affect overall operations of the Marie Curie rover. These include:

- ♦ Earth-Mars one-way communications time delay (5-20 minutes)
- ♦ Limited communications bandwidth (generally < 10 Mbits downlink per sol[1] available to rover)
- ♦ Limited communications opportunities (1 command uplink, 2 telemetry downlinks per sol)

The power system is the single most important resource for the Marie Curie Rover. This system consists of a .22 square meter solar array and 9 LiSOCL batteries. The batteries on Marie Curie are primarily used during the night for APXS data collection. They are primary batteries and therefore modeled as non-renewable depletable resources. The solar array is the primary power source used during the day. The predicted available solar power profile throughout the Mars day must be input before planning begins. Using a daily model is required due to changing solar array power available as a result of degradation from dust accumulation and seasonal solar irradiation variability. The angle of the solar array, which depends on the terrain, will also affect the availability of solar energy. Solar array angle estimates could be generated by RCW for input into ASPEN.

A typical Mars day might involve a subset of the following activities:

- ♦ Complete an APXS data collection that was carried out during the prior night
- ♦ Capture a rear image of the APXS site
- ♦ Traverse to an appropriate site and perform a series of soil mechanics experiments, including several subframe images of soil mounds and depressions created by running individual wheel motors
- ♦ Traverse to a designated rock or soil location
- ♦ Place the APXS sensor head
- ♦ Capture end-of-day operations images with its forward cameras
- ♦ Begin APXS data collection
- ♦ Shut down for the night

APXS data collection usually occurs overnight while the rover is shutdown. Each of these activities can be input into ASPEN as a goal for that Mars day planning horizon. The format of the input goals is RML or Rover Modeling Language. RML is an application of Extensible Markup Language (XML) designed specifically for rover operations. RCW will use RML for input and output. RML is described in detail in the next section of this paper.

The exact position of the rover after a traverse activity is subject to dead reckoning error. The timing of traverse activities is also non-determinant. Because of the inherent problems of coordinating activities between the event-based rover and time-based lander, wait commands are used to synchronize activities. When the lander is imaging the rover after a traverse, a wait command is used to ensure the rover will remain stationary at its destination until the lander completes imaging. Because the rover executes commands serially, this ensures that another command will not start execution before the previous command has completed. All rover traverse goals are generated using

[1] A Sol is a Martian day, equivalent to about 24 hours and 39 minutes

the RCW. (ASPEN is not designed to perform rover motion planning.) The RCW operator can fly a 3-D rover icon through the stereoscopic display of the Martian terrain. By inspecting the stereo scene, as well as placing the rover icon in various positions within the scene, the operator can assess the trafficability of the terrain. By placing the icon in the appropriate position and orientation directly over the stereo image of the actual rover on the surface, the rover's location and heading are automatically computed. This position information is output to ASPEN to set the rover end position state. The rover driver specifies the rover's destinations by designating a series of waypoints in the scene, generating waypoint traverse commands.

Rover data storage is a scarce resource that must be tracked within the ASPEN model. The largest consumer of data storage is the camera image activity. This activity can fill the on-board data storage if a telemetry session with the lander is not available during the data collection. ASPEN will keep track of the data storage resource to ensure that all data is downlinked before the buffer is completely full.

Planning Tool Interfacing: Rover Markup Language (RML)

There are several different tools that can be used for developing rover sequences. In addition to ASPEN and RCW, other tools can be used for environmental predictions, distributed science planning, instrument analysis, and engineering performance analysis. Each of these tools is created by a different set of engineers or scientists that are cognizant in that particular piece of the rover operation. In order to simplify the interface between these rover tools, we decided early on to use a common interface language. We needed to capture all information about rover command generation and uplink, preferably in exactly one file per uplink. This information includes the following:

- ♦ The science requests that the uplink is designed to satisfy, and the originator of each request
- ♦ The rover commands to be uplinked, with each command cross-referenced to the request or requests it helps to implement
- ♦ The operators who worked on the uplink
- ♦ The downlink telemetry related to the uplink
- ♦ References to auxiliary files, such as terrain databases, that were used in preparing the uplink

In addition, we must generate uplink and downlink reports, preferably in HTML, so that we can post them on web sites accessible to the operations teams. The Pathfinder team created these reports manually, requiring several hours of tedious work for every uplink; we wanted to capture all the information needed for these reports in the file, so that future missions could generate the reports automatically.

We chose to base our data language on XML for several reasons. First, XML is an emerging data representation standard with widespread support from both proprietary-software and free-software organizations. Because XML is free and open-source, there is a wide community of users supporting development of tools and utilities that make XML easier to use. Included in this set of tools are numerous free, high-quality parsers usable from several programming languages. We didn’t have to design a data language from scratch (and document it) and then write, test, document, and maintain a parser for it. All that was necessary was to download a free parser, plug it in, and run it. Because all of the XML parsers expose a standardized API, we can switch parsers with a minimum of effort and no changes to data files if a better implementation comes along.

Because XML parsers are available for several languages, we can use the right language for each job. We can write larger applications in languages such as C++ or Java, and smaller applications (e.g., the HTML report generators) in languages such as Perl, Python, or Tcl. All of these languages can parse our XML data equally well, with no extra effort on our part.

XML files tend to be naturally modular. As we discover the need to capture an additional datum, it’s usually trivial to add a section for it into our evolving specification. Also, because the external representation for XML is based on ASCII, standard Unix shell tools and text editors work with it. We can search for the existence of particular tags, for instance, or quickly develop a test-input file using any standard text editor.

We still have to do some work to design and document our subset of XML. We also have to perform some data verification. For example, you can’t tell XML parsers to insist that a field’s value be a sequence of digits, for instance, so we have to write our own code for that. Still, XML gives us a great base to start from and a great choice of existing tools, saving us a lot of time, money, and labor.

```
<Commands>
        <CMD__waypoint__>
            <ARG__waypoint____X>1015</ARG__waypoint____X>
            <ARG__waypoint____Y>1433</ARG__waypoint____Y>
            <ARG__waypoint____time>1</ARG__waypoint____time>
            <Satisfies>photo</Satisfies>
        </CMD__waypoint__>
        <CMD__turnheading__>
            <ARG__turnheading__>9830</ARG__turnheading__>
            <Satisfies>photo</Satisfies>
        </CMD__turnheading__>
        <CMD__wait__greater>
            <ARG__wait__greater__sensor>SEN_ISOLAR</ARG__wait__greater__sensor>
            <ARG__wait__greater__value>42</ARG__wait__greater__value>
            <ARG__wait__greater__limit>15</ARG__wait__greater__limit>
            <Satisfies>photo</Satisfies>
            <Comment>Wait up to 15 minutes for decent lighting</Comment>
        </CMD__wait__greater>
        <CMD__image__>
            <ARG__image____shift>shift0</ARG__image____shift>
            <ARG__image____camid>left</ARG__image____camid>
            <ARG__image____time>11</ARG__image____time>
            <ARG__image____compression>btc</ARG__image____compression>
            <ARG__image____apid>0</ARG__image____apid>
            <ARG__image____srow>1</ARG__image____srow>
            <ARG__image____scol>0</ARG__image____scol>
            <ARG__image____erow>256</ARG__image____erow>
            <ARG__image____ecol>256</ARG__image____ecol>
            <Satisfies>photo</Satisfies>
        </CMD__image__>
</Commands>
```

Figure 5 - Rover Markup Language Example

Figure 5 contains an example of RML. This example consists of commands to take a picture with the left front rover camera. Included are commands to turn to the photo target, wait for proper lighting, and take the picture with the proper camera parameters. These commands are part of a “photo” request defined in RML. Information about the requestor is also encoded in RML but not shown in this example.

Status

Initial work in 1998 consisted of a preliminary proof of concept demonstration in which we used automated planning and scheduling technology integrated with WITS to demonstrate automated commanding for the Rocky-7 rover from the WITS interface. (Backes, et al., 1999) In 2000, we are providing an in-depth validation of the automated command-generation concept. The ASPEN planning and scheduling system will be integrated with a rover activity interface and the Rover Control Workstation. ASPEN will receive RML formatted high-level requests from the activity interface. ASPEN will then automatically generate validated rover-command sequences that satisfy these requests and provide those RML formatted sequences to the Rover Control Workstation. The ASPEN Java-based interface will enable the user to access planned activities and to observe resource and state constraints. As the ASPEN interface is Java-based, users will be able to access this commanding capability from anywhere on the Internet. The computation intensive aspects of the commanding capability (such as the planner/scheduler, path planner, uncertainty estimation software, vision and image processing software, etc.) will reside on one or more rover workstations based in a central location.

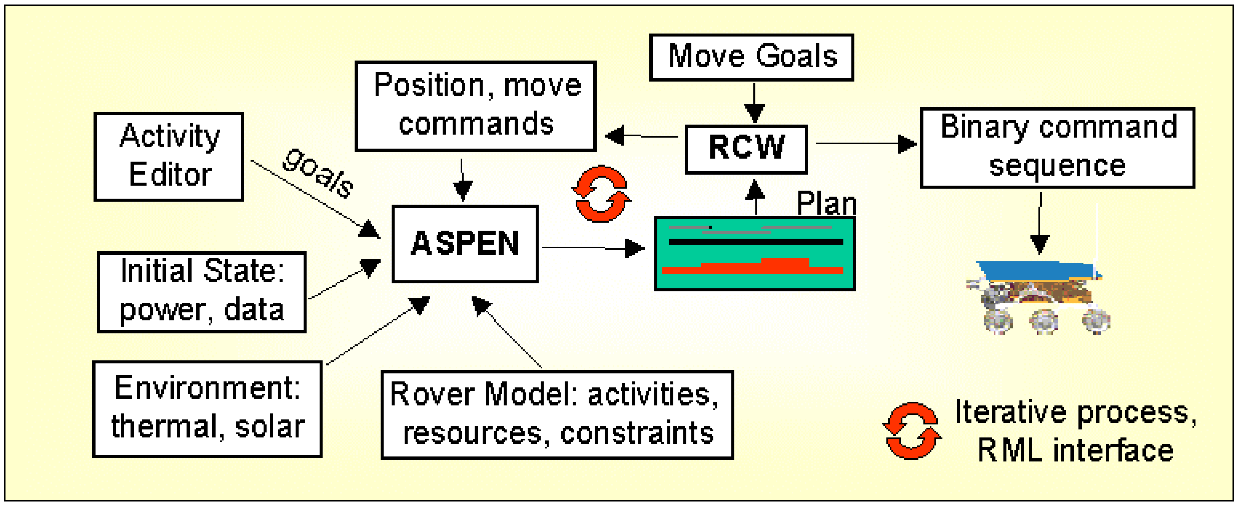
Move Goals
Position, move commands
Activity Editor
goals
RCW
Binary command sequence
ASPEN
Plan
Initial State: power, data
Environment: thermal, solar
Rover Model: activities, resources, constraints
Iterative process, RML interface

Figure 6 - End-to-End Automated Commanding System

The end-to-end data flow for this system is shown in Figure 6. The interaction between ASPEN and RCW is an iterative process. Both ASPEN and RCW will receive high-level goals. The RCW input goals will be related to rover motion. RCW will output traverse commands for input into ASPEN. ASPEN will merge these with other science and engineering goals to produce an intermediate level plan. The plan will be output to RCW to update motion commands as necessary. This process will continue until an acceptable plan is generated. Finally a time ordered list of commands would be output for sequence generation.

The Marie Curie ASPEN model is nearly complete and ready for testing. Initial testing on a sample of 136 activities produced a conflict free plan in about 9 seconds. This testing was completed on a Sun Ultra-2 workstation. These relatively quick plan cycles will allow the Marie Curie Rover operations team to perform "what-if" analysis on different daily plans. Our goal is that this quick planning capability will be used to generate commands more frequently than once-per-day, if communications opportunities permit.

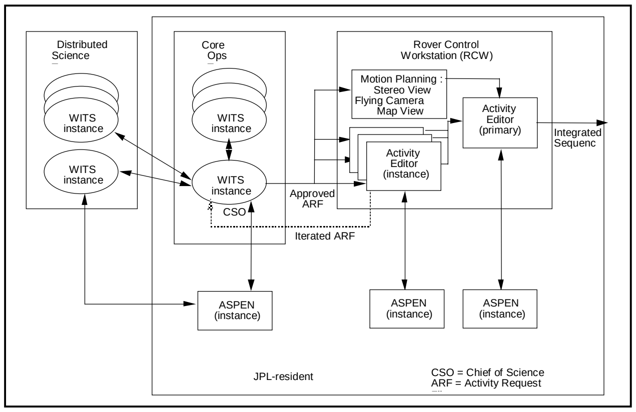

Figure 7 - Possible Rover Uplink Dataflow

Our next level of testing will involve generating plans for two typical Sojourner rover days on Mars. These plans will be compared with the manually generated sequences that were run during the Sojourner mission. As a result of these tests, minor updates to the model may be required. Once the model is validated, we will integrate ASPEN with RCW. Figure 7 shows a possible Marie Curie rover uplink operational data flow. The highlighted boxes show the planner that would be used at both the science planning and engineering planning level. The planner model would contain sufficient engineering information to ensure that the vast majority of science requests finally approved are feasible from an engineering standpoint. Eventually we would like to add performance metrics to the planner model to optimize the generated plans. This will enable automated "what-if" analysis to generate plans that maximize science and engineering value.

Onboard Rover Planning

In addition to the work with Marie Curie, we are developing a dynamic, onboard planning system for rover sequence generation. The CASPER (Continuous Activity Scheduling, Planning, Execution and Re-planning) system (Chien et al., 1999; Chien et al., 2000), is a dynamic extension to ASPEN, which can not only generate rover command sequences but can also dynamically modify those sequences in response to changing operating context. If orbital or descent imagery is available, CASPER interacts with a path planner to estimate traversal lengths and to determine intermediate waypoints that are needed to navigate around known obstacles.

Once a plan has been generated it is continuously updated during plan execution to correlate with sensor and other feedback from the environment. In this way, the planner is highly responsive to unexpected changes, such as a fortuitous event or equipment failure, and can quickly modify the plan as needed. For example, if the rover wheel slippage has caused the position estimate uncertainty to grow too large, the planner can immediately command the rover to stop and perform localization earlier than originally scheduled. Or, if a particular traversal has used more battery power than expected, the planner may need to discard one of the remaining science goals. CASPER has been integrated with control software from the JPL Rocky 7 rover (Volpe et al., 2000) and is currently being tested on Rocky 7 in the JPL Mars Yard.

Conclusions

Current approaches to rover-sequence generation and validation are largely manual, resulting in an expensive, labor and knowledge intensive process. This is an inefficient use of scarce science-PI and key engineering staff resources. Automation as targeted by this system would automatically generate a constraint and flight rule checked time ordered list of commands and provides resource analysis options to enable users to perform more informative and fast trade-off analyses. Initial tests have shown planning times on the order of seconds rather than hours. Additionally, this technology would coordinate sequence development between science and engineering teams and would thus help speed up the consensus process.

Enabling goal-driven commanding of planetary rovers by engineering and science personnel greatly reduces the workforce requirements for highly skilled rover engineering personnel. The reduction in team size in turn reduces mission operations costs. In addition, goal-driven commanding permits a faster response to changes in rover state (e.g., faults) or science discoveries by removing the time consuming manual sequence validation process, allowing "what-if" analyses, and thus reducing overall cycle times.

Acknowledgement

The TMOD Technology Program funded the research described in this paper. The Mars Surveyor Operations Office, the Athena Rover Office, and the Athena Precursor Experiment funded the RCW development work. All work was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Bibliography

J. Bresina, K. Golden, D. Smith, and R. Washington, Increasaed Flexibility and Robustness of Mars Rovers, Proceedings of the 5th International Symposium on AI, Robotics, and Automation in Space, Noordwijk, The Netherlands, June 1999.

P. Backes, K. Tso, and G. Tharp. "Mars Pathfinder mission Internet-based operations using WITS. In Proceedings IEEE International Conference on Robotics and Automation," pages 284-291, Leuven, Belgium, May 1998.

P. Backes, G. Rabideau, K. Tso, S. Chien, "Automated Planning and Scheduling for Planetary Rover Distributed Operations," Proceedings of the IEEE Conference on Robotics and Automation (ICRA), Detroit, Michigan, May 1999.

S. Chien, R. Knight, A. Stechert, R. Sherwood, G. Rabideau, "Integrated Planning and Execution for Autonomous Spacecraft", Proceedings of the IEEE Aerospace Conference (IAC), Aspen, CO, March 1999.

S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using Iterative Repair to Improve Responsiveness of Planning and Scheduling," Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling, Breckenridge, CO, April 2000.

S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D. Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, D. Tran , "ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling," SpaceOps 2000, Toulouse, France, June 2000.

A. Fukunaga, G. Rabideau, S. Chien, D. Yan, "Toward an Application Framework for Automated Planning and Scheduling," Proceedings of the 1997 International Symposium on Artificial Intelligence, Robotics and Automation for Space, Tokyo, Japan, July 1997.

A. Mishkin, "Field Testing on Mars: Experience Operating the Pathfinder Microrover at Ares Vallis," presentation at Field Robotics: Theory and Practice workshop, May 16 1998, at the 1998 IEEE International Conference on Robotics and Automation, Leuven, Belgium.

A. Mishkin, J. Morrison, T. Nguyen, H. Stone, B. Cooper, B. Wilcox, "Experiences with Operations and Autonomy of the Mars Pathfinder Microrover," proceedings of the 1998 IEEE Aerospace Conference, March 21-28 1998, Snowmass at Aspen, Colorado.

G. Rabideau, R. Knight, S. Chien, A. Fukunaga, A. Govindjee, “Iterative Repair Planning for Spacecraft Operations in the ASPEN System,” International Symposium on Artificial Intelligence Robotics and Automation in Space (ISAIRAS), Noordwijk, The Netherlands, June 1999.

R. Volpe, T. Estlin, S. Laubach, C. Olson, and J. Balaram, "Enhanced Mars Rover Navigation Techniques" To appear in the Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, April 2000.

Zweben, M., Daun, B., Davis, E., and Deale, M., “Scheduling and Rescheduling with Iterative Repair,” Intelligent Scheduling, Morgan Kaufmann, San Francisco, 1994, pp. 241-256.