science return that may not have been possible otherwise.

References

Bresina J., Morris P., "Mission Operations Planning: Beyond MAPGEN", *Second IEEE International Conference On Space Mission Challenges for Information Technology (SMC-IT)*, Pasadena, 2006, To Appear.

Bresina J., Jónsson A., Morris P., Rajan K. "Mixed-Initiative Planning in MAPGEN: Capabilities and Shortcomings." *Workshop on Mixed-Initiative Planning and Scheduling*, ICAPS05, 2005.

Bresina J., Jónsson A., Morris P., and Rajan K., "Activity Planning for the Mars Exploration Rovers", *Fourteenth International Conference on Automated Planning and Scheduling*, Monterey, 2005, pp. 40-49.

ADDITIONAL GOOD TEXT

The barrier to adoption of any new software system can be related to the difficulty of learning and interfacing with that system. In addition, when an automation related component that might introduce any degree of risk into a mission is suggested for inclusion, a considerable degree of trust from the mission must exist before that component has any possibility of being relied upon. A planner in particular is considered particularly complex in the sense that a user will not necessarily understand the process it takes to arrive at its decisions. As it is not necessarily trivial for the planner to explain its reasoning in a human understandable fashion, an additional degree of risk is perceived even beyond something like an automation task where risk might include something simple like whether a timed event occurs on schedule or not. When the user does not understand why a software component performs some action that increases the amount of trust required for adoption. Trust can be gained over time by a mission using software in a trial environment.

MER

Operation of the two MER rovers, Spirit and Opportunity, has continued far past their designed ninety day primary missions. < JPL partners in Ensemble have deployed a new system for browsing images downlinked from the rovers,

Phoenix

The Phoenix Mars lander launches for Mars in August 2007, and is scheduled to begin operations in the north polar region of Mars in early 2008. The goal of the Phoenix mission is to understand the chemistry and water cycles in the Martian northern latitudes. Among other activities, Phoenix will scoop samples of the Martian soil and frost into an onboard chemistry lab for analysis.

<PSI – includes Maestro, SPIFe, and Europa 2. Interestingly, Europa was not originally baselined for inclusion in PSI due to the tight budget and schedule of the PHX mission. As we developed the Europa 2 server and Ensemble plugins for MSL, it became natural to demonstrate it and eventually deploy it for PHX. On PHX, we do not make use of all capabilities of Europa 2, as APCORE, a JPL plan analysis tool, plays some of that role>

MSL

This is example text.

Advantages of the Approach

- Another major feature that missions like to see is the ability to use a particular piece of software across missions. Multi-mission software reduces the long term cost and once people are familiar with it there is a major advantage to additional missions using it both training and trust wise. Low cost incremental improvements can be applied to software to benefit from lessons learned on each mission while still leaving the core software.
    - system engineering between missions
        - we participate in system engineering in multiple missions, so techniques, components used for one can flow across
-
    - builds confidence
    - reduced cost
- incrementality
    - builds confidence
    - lowers risk
    - missions can easily evaluate/add components being developed for other missions
        - Example: Europa on PHX
- flexibility in process
    - allows our tools to fit into process
    - allows non-linear process
- rapid feedback
    - cannot stay on the wrong path for very long due to user feedback, constant demos

Risks That Must Be Managed

- mismatched requirements between missions, common approach may be least common denominator
- edits to one project may impact/break others
- having multiple missions makes PM more difficult
- failure to take ownership --funding

Concluding Remarks

Mission systems have a certain amount of inertia and it can be difficult to change a system for the better even when the users know it would be beneficial. It is important to remember that this is as much of a social challenge as a technical one. One of the most important lessons learned throughout this process has been that of recognizing a user and process centric solution as having more success than a purely technical one. By working within the existing process and slowly introducing new technical aspects as previous ones gain the acceptance and trust of the mission, components that have a higher degree of risk involved can be suggested.

Improvements in process by analyzing how software is used and then designing software based on that interaction rather than as an ad hoc integration of technological components can provide a compelling motivation for adoption of a system. That said, the ease of technical integration cannot be underestimated. A “come one, come all” approach and being willing to interface to any component brought to the system makes it more difficult to argue against adoption of the system.

Extended mission operations provide a forum for somewhat riskier software to participate in a mission. Planner usage that may not be even considered during primary operations could one day become essential to running an extended mission with reduced staffing while providing a

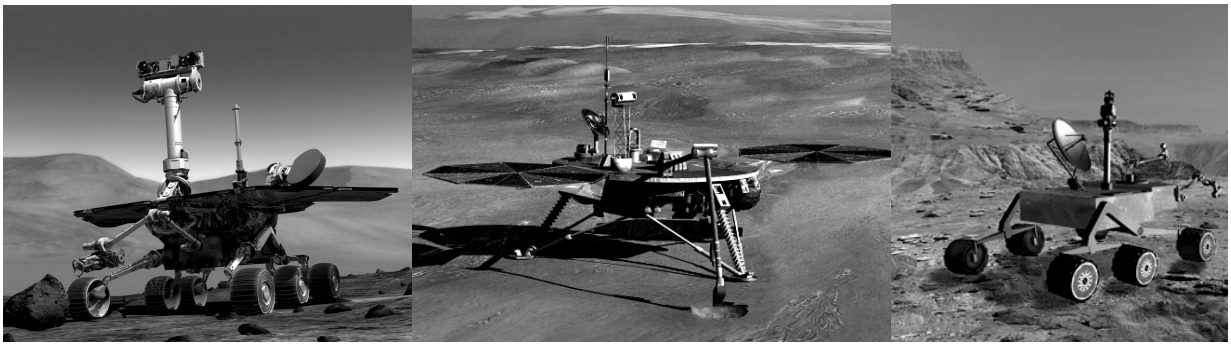


Figure 1: MER rover, Phoenix lander, and MSL rover

Simple, agnostic interfaces

- no highly customized planner application per mission
- standard API
- custom model generated from existing mission artifacts
- small customization

One way to lower the barrier to adoption is to decrease the complexity of implementation. This also lowers the perceived risk to the mission if they later decide to drop a particular feature. A small investment is easier to approve than a major system integration task that they are too heavily entrenched in to remove later if needed. As it is unknown what platform, language, or architecture is used by any particular mission system, it is necessary to devise a method by which most any system can interface with the new software component. The method chosen which implements such agnostic interfaces is a network component architecture, specifically known as web services in this context.

XML-RPC, a relatively simple to use and widely implemented protocol was chosen as the basis for the interface between architecture components. This lightweight remote procedure call protocol uses XML to encode requests and responses and uses HTML as the transport mechanism. XML-RPC libraries are available for a wide variety of languages which makes it ideal for easy adoption by a prospective mission. The simple function based API requires a minimal amount of time to understand before implementation can proceed.

The API developed consists mainly of operations related the to adding, modifying and deleting activities and constraints and modifying activity parameters. Once the activities and constraints are registered, the client can call the various functions to determine the state of the network or the activities, such as network consistency, getting a list of temporal and flight rule violations, or fixing those violations. The interface also allows for activating or deactivating flight rules and the ability to switch between active or passive enforcement of flight rules.

- PA written at Ames. Ames and non-Ames planners can connect
- we've turned the DS1 idea inside out
    - On DS1, the planner was connected to planning experts – services that could provide is specialized info on navigation, etc.
- In Ensemble, the planner is a set of specific services that can be called by other applications
-

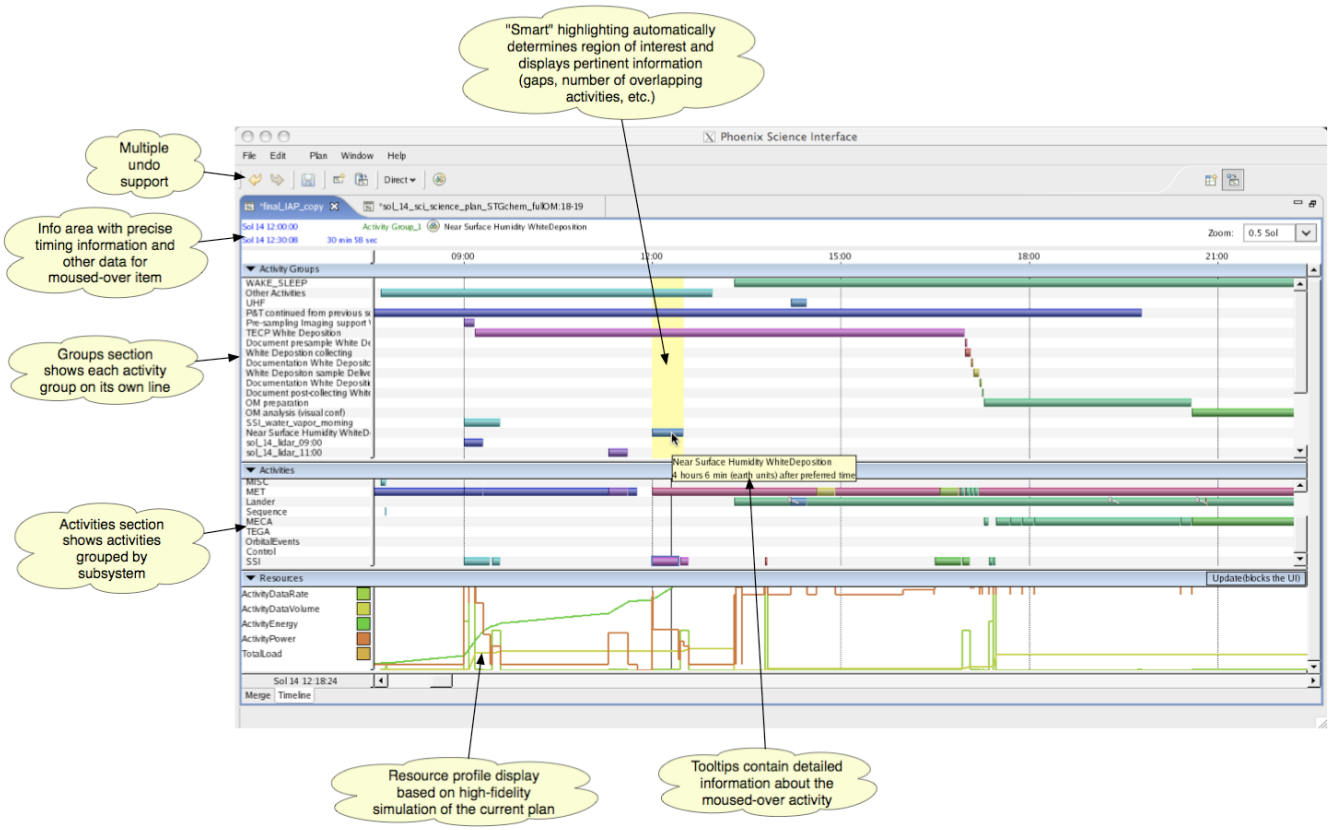
Models generated from existing mission artifacts

A custom model of the spacecraft activities is required to gain the full benefit of the Europa planner. A first pass interface can be used without a model by using a generic model developed which allows the use of generic activity types. This only allows the planner to perform temporal constraint problems but is still useful for rapid adoption by allowing an immediate benefit to a mission to be derived with a minimum of effort.

An experimental language was developed [Bresina] to define the activity dictionary and to then automatically convert this to the model used by Europa. This allowed a very lightweight dictionary to be used as a single source data model for both the client and the back end planner. While the experimental version had some specific assumptions that made it fairly domain specific this concept could be expanded to make it even easier to produce models automatically and insure synchronization between multiple components in an architecture.

The Missions (very draft)

Steal text from IS infusion report



Maestro Just a quick description and pointer to Maestro papers, as this audience will think of "planning" as more of what Europa does.

Science Planning Interface (SPIFe) The image above is SPIFe, with data being provided by a number of specialized services, including the Ames Europa 2 planner and APCore, a JPL plan modeling system.

Europa 2 Server This is example text.

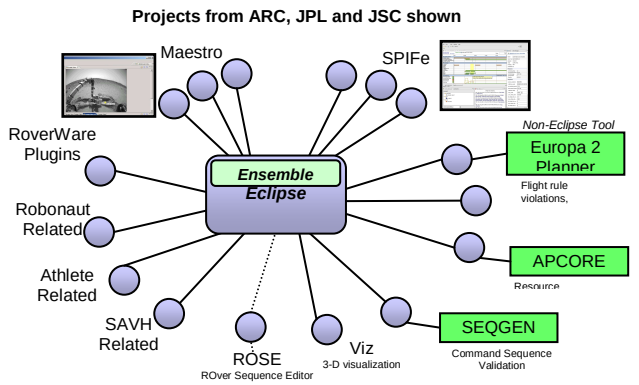
Eclipse

The Eclipse platform is an essential tool for development within Ensemble. The benefits of extending Eclipse for use in Ensemble applications are:

- By using a toolkit that provides an API to handle window management, file IO, dialog creation, developers can focus upon their core competencies.
- Consistent look and feel and metaphors across the entire application
- Cross platform with the added advantage of native UI libraries to enhance the look and feel above that of Swing based applications

Additional benefits are gained through the use of the extension point model in Eclipse. The extension point mechanism allows for loose coupling of software components to each other. They behave as the contracts between one component that defines an extension point (such as a file menu that allows for additional menu item by requesting information regarding the name, icon and action to be executed) and a component that implements it. This simple concept can be extended to provide a great deal of information that may be optionally included in the product or left out, and leads to Eclipse's plug-in architecture that provides Ensemble with the flexibility necessary to mix and match features for particular missions.

One dramatic example of this mechanism are the perspective extension points. Perspectives in Eclipse are visual containers that encapsulate views, editors, menus and fundamentally the operation of the entire application. Using a perspective a user can rapidly switch back and forth from the data browsing perspective to the planning and scheduling perspective with the click of a button. This functionality essentially melds the functionality of two entirely different applications into a single desktop executable. Such capabilities further allow for great interoperability between the applications because cooperation is no longer dependent upon external interfaces (file IO, network communications, etc), but can instead implement application programming interfaces (APIs) that are far more flexible and robust.



Open Component-based Development

A second key observation from MER operations is that for a mission with tight time requirements on the tactical planning process, lack of integration between operations tools poses significant efficiency problems. The MER operations process, along with that of many other missions, was assembled from many existing tools. Typically, experts on each step of the operations process described above develop a specialized tool with its own user interface, data structures and formats, and way of conceptualizing the operations process. These are then assembled into the operations process by adapting them to each other with translators, scripts or other means of integration.

During operations, this means that users must move from tool to tool to perform the operations flow,

Ops-time issues;

- Not designed to be part of same conceptual process
- Duplicated, inconsistent interfaces to same subtask
- Translations between components one-way, so no going back

Development issues

- Tools are not integrated until very late, and in a more ad-hoc manner. Not clear when it's broken
- Time is wasted developing incompatible and redundant data formats, displays etc. Later more time is wasted integrating them together.

Ensemble is a software platform for operations which allows software from different teams to be integrated together into a single ops application. Ensemble is based on components, meaning different software functions are delivered as re-usable components that can be combined together depending upon what type of application is needed. We are currently delivering overlapping but unique applications to three missions and a host of smaller research projects. The figure above gives a flavor of the number and variety of teams involved to date.

Core parts of ensemble

- Shared data structures to represent plans and lower level command sequences, checked at compile time.
- Shared implementation of routine tasks (io, etc)
- Shared, cooperative design and implementation of cross-cutting functionality
- Method of integrating

A variety of software components for different tasks — from manipulating scientific images to validating complex sets of spacecraft activities — have been plugged into Ensemble. The Ensemble platform is heavily based upon Eclipse, an open-source, component-based Java software platform, which we discuss briefly in the next section.

In addition to the client side plug-in components, maestro also uses network components to gain additional functionality. Either for convenience of integration, architectural elegance, or for enhanced security, it sometimes makes sense for certain components to reside as back end server processes and communicate with the client via some network protocol.

Computationally intensive components such as planners can benefit from residing on appropriately sized servers that can perform reasonably without being bounded by the client machine the front end runs on. This allows users to gain the benefit of a planner without becoming frustrated by a lack of responsiveness they might otherwise expect.

Ensemble is also a software development process by which many teams from different centers can cooperate to develop mission operations software.

- Shared code and documentation repositories
- Early integration of components
- Continuous automated build and unit test
- rapid development process + prototypes

- The planning software itself is more correctly described as a system of specialized, interconnected tools that each perform a subset of operations, e.g must rely on other tools for creating the activities we plan upon, estimating their resource usage, etc
- The software ecosystem changes from mission to mission

In the remaining sections of the paper, we discuss how the approach to design and development of planning tools differs from that used on MER, including heavy use of Human/Computer Interaction expertise and the integration of the planner engine within a multi-mission, component-based software tool set. We briefly described the missions where we are currently employing this approach, the advantages we have seen and the accompanying risks that must be managed.

Approach

Based on observation of MER operations, experience of team members developing tools for that mission, and feedback from MER users, we take an approach that differs somewhat from MER planning tool development. These differences can be found at all stages, from how we decide what it is we are building, to how we design and develop it, to how we deploy it. We describe a number of these below.

Use of Human/Computer Interaction Expertise

An advantage of following the highly successful MER mission is that we can study and learn from that experience. Planning experts and software engineers from our current team were on the team that developed the MAPGEN and Constraint Editor tools for the MER mission, and participated in operation of the primary ninety day mission for the two MER rovers. In addition, a team of Human/Computer Interaction (HCI) experts from NASA Ames studied MER operations for ninety days. Their focus was to understand the MER operations process and how the entire human and software system operated to produce a plan and command load every day, independent of the role of any particular technology.

When the opportunity came to develop a streamlined activity planning system for subsequent Mars surface missions, the HCI team developed a prototype of an integrated activity planning and scheduling application that showed how bottlenecks and frustrations encountered during MER operations could be reduced or eliminated. This helps to ensure that tools and technologies are adapted to support the best possible process, rather than a process being developed to fit an amalgam of the available technologies, and made it clear to subsequent missions that we could hear and respond to their highest priority problems. To this date, the HCI team remains an equal partner in delivering operations tools to missions by

<EDIT THIS>

- Working with software engineers and planning experts on a weekly basis to determine what features and technologies will best
- Story-boarding how users would interact with the system
- Performing structured user tests to gather early feedback and determine best designs and processes

User Control

A key design element in the architecture of the system based upon MER feedback and HCI studies is that of user control. When creating or modifying a plan, the user must have almost complete control over where activities are placed on a timeline. It can be very disconcerting for activities to jump around for an apparently unknown reason, even if it is to correct a temporal or flight rule violation. It is therefore important that the user know why a particular action is taken place.

In the case of the current design, the user is essentially given complete control and uses the planner as an advisor. As the user manipulates the plan, the system is automatically communicating with the Europa planner behind the scenes, determining if there are temporal violations and allowing features like constrained move where the interface will move constrained activities within their allowed temporal bounds as the user moves a target activity. The system will automatically generate a list of violations that exist for various activities and attempt to explain what is causing the violation. The user can manually select “Fix Violations” to have the Europa planner attempt to fix all the existing violations in the plan.

The main point is that the planner is almost always acting in a passive way. It will alert the user to a violation without acting upon it unless explicitly told to do so by the user. The planner does not create user level activities or constraints on its own. It will suggest such actions to the user in appropriate situations but then relies on the client to manipulate the plan according to those suggestions at the user’s discretion.

Work in a reference to Europa 2 changes that allow us to do passive checking. Reference Bresina & Morris SMC-IT paper.

planning in this context that are different than from how planning problems have typically been framed in the computer science community. We suspect these differences may be relevant to a number of similar types of operations processes. We then discuss some of the changes in approach we have taken, from how we decide what functionality to include in a planning application, to how we design and implement it, to how we develop the planning models. We briefly discuss the ongoing deployment of a planning application across the PHX, MSL and potentially MER missions. Based on this experience, we introduce some of the advantages we feel this approach has allowed us, and the risks it introduces.

Operation of a Surface Mission

When scientists and engineers operate a lander or rover on a planetary surface such as Mars or a moon, they typically have a strategic plan that lays out what kinds of activities they would like the rover to perform over the next few days or weeks. With today's highly capable spacecraft, many of these activities will involve complex interactions with the environment, such as driving through rough terrain, digging, heating core samples, or in the near future, ablating rocks with a laser. The precise outcome of these activities, from how much energy will be consumed driving to a destination, to what scientific opportunities will be revealed by digging and photographing a trench, is difficult to predict in advance. Thus mission operations will typically consist of a longer term strategic planning process, and a tactical planning cycle that takes into account the problems and opportunities encountered on the surface each day, referred to as a sol on other planets.

The tactical commanding cycle might proceed as follows. The engineering and science data gathered the previous sol is analyzed to determine the status of the rover and its surroundings. This might include images taken by the rover, non-image science data such as spectra, readings from engineering sensors on the rover, and reports on the state of batteries or data storage devices.

Based on this, and guided by the strategic plan, the scientists determine a set of desirable scientific objectives for the next sol. To do this, they might browse downlinked images and data to determine what objects are within the range of the rover's sensors and actuators. They might decide for example that a specific point on a local rock is of interest for drilling or that features in the distance should be imaged. For each activity, there will be a number of science-driven parameters, such as which filter should be used when taking an image, and science-driven constraints, for example to ensure that an image is taken at a time of day when the lighting is adequate. In addition to setting parameters and constraints, scientists must take care that their desired objectives are not consuming an inordinate amount of the rover's shared power and data storage resources. Since a detailed picture of the resource usage of all scientists' desired activities is not yet available, the scientists are encouraged to oversubscribe somewhat to ensure that the rover's resources will be fully utilized.

In the next step in the tactical process, the observation requests from all of the scientists must be merged with each other and with engineering requirements (e.g., the rover may have to remain stationary at a fixed time of day to relay its data to a satellite passing overhead). This merging process may require deletion or modification of some activities, with accompanying negotiation between scientists, trading for more access to the rover on subsequent sols, and so on. From the merged set of requests, a detailed plan of activities is constructed for the upcoming sol. The plan must obey all applicable flight rules that specify how to safely operate the rover and its instrument suite, must remain within specified resource limitations, and should obey the science constraints placed upon each activity. Moving from the requested set of observations to a valid plan may again involve removing or modifying activities or deleting or adjusting science constraints, with the requisite negotiations. The resulting activity plan is then reviewed and approved by the scientists and engineers.

Once approved, the activity plan is used as the basis to create sequences of spacecraft commands, which drive onboard execution. This sequence structure is then validated, packaged, and communicated to the rover. The rover executes the sequences, downlinks the resulting data, and the tactical cycle begins again. Depending upon the nature of the mission, the tactical cycle may repeat on the order of hours or days.

There are many points in the strategic and tactical planning processes where planning software is helping the MER mission, and will help the Phoenix and MSL missions. It's interesting to note however that there are many important aspects of these mission deployments that wouldn't necessarily come to mind when thinking about planning technology:

(Need to edit/add to these to make them foreshadow the approach/advantages we have later)

- Human negotiation is used to determine which goals should be in or out of a plan when
- Users must be able to work with inconsistent plans, efficiently implement negotiated changes to the plan or constraints, and explain the resulting plan to their colleagues
- the planning software operates within a large software ecosystem created by many specialized teams often working in isolation until system integration

Planning Applications for Three Mars Missions with Ensemble

Arash Aghevli[+], Andrew Bachmann[+], John Bresina[*], Kevin Greene[+], Bob Kanefsky[#], James Kurien[*], Michael McCurdy[*], Paul Morris[*], Guy Pyrzak[^], Christian Ratterman[^], Alonso Vera[*], Steven Wragg[+]

[*]NASA Ames Research Center
[+]QSS Group, Inc.
[^]San Jose State University
[#]University of California
MS 269-2, NASA Ames Research Center, Moffett Field, CA 94035
<preferred email addresses>, James.A.Kurien@nasa.gov, <preferred email addresses>

NOTE: This is a draft. We will submit another by the end of the day July 5, but are submitting this in case we have any problems later.

Abstract

A number of new tactical planning and operations tools were deployed on the highly successful Mars Exploration Rover (MER) mission. Based on successes and lessons from the MER experience, a number of groups at NASA Ames and JPL have developed a platform for developing integrated operations tools, called Ensemble. Ensemble is a multi-mission toolkit for building activity planning and sequencing systems that is being deployed on extended operations for the MER mission, the 2007 Phoenix Mars Lander and the 2009 Mars Science Laboratory rover mission. This paper begins with an overview of typical surface mission operations. After an introduction to Ensemble, this paper discusses the components of Ensemble that focus upon what the planning and scheduling community would consider the planning portion of Ensemble. We discuss the deployment of a planning application across the PHX, MSL and potentially MER missions, some salient features of our design, development and deployment process, how those features enable us to get on the missions and support multiple missions, and what risks they introduce.

Introduction (very draft)

The Mars Exploration Rover (MER) mission is a highly successful rover mission to Mars run by NASA's Jet Propulsion Laboratory (JPL). Each day scientists and engineers must analyze what situation the two MER rovers are in and plan what activities the rovers will perform the following day. To support this process, a number of planning tools were developed by teams at JPL and the NASA Ames Research Center, including a tool scientists use to define the activities they would like the rover to perform (e.g., grind the surface of a specific rock for 30 minutes) [ref SAP], a tool used to create constraints between activities (e.g., the rock grinding must take place before the picture taking activity that documents it) [ref CE], and a mixed-initiative planning tool used to assemble the activities into a plan that is consistent with the constraints, safe operations rules for the rovers, and limits on power and other resources. These tools have been used for over two years on the two MER rovers, resulting in the successful production of nearly two thousand plans as of this writing.

The MER mission used planning technology in tactical operations in a way that had not been done before. At the end of the ninety day primary MER mission, a colleague quipped that the best way to develop software for a mission was to fly the mission first, then write the software, because at least then the requirements would be known. In some sense, the authors have that luxury as part of a team responsible for deploying activity planning and scheduling tools for the 2007 Phoenix Mars Lander and the 2009 Mars Science Laboratory (MSL) rover mission. The Phoenix and MSL missions are very different from MER in of scientific objectives, instrument packages and spacecraft capabilities. But we have a model of how planning technology was used in MER operations, and lessons learned from developing it, integrating it, and participating in real mission operations. Moreover, during MER primary operations, a Human/Computer Interaction (HCI) team studied the MER tactical process to better understand what task the combined human/software process was attempting to achieve each day, how people were actually using the planning technology that was deployed, and how they would have liked to have use it.

All of these lessons have lead us to build on the success of these tools with a somewhat different approach to designing and deploying applications with planning technology which we believe will make future applications even more efficient to use and easier to implement. This experience may be of use and interest to people working on similar kinds of applications, space related or not.

This paper begins with an overview of typical surface mission operations and notes some characteristics of