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
<preferred email addresses>, James.A.Kurien@nasa.gov, <prefe

This is a somewhat rough account of how our personal view of deploying planning technology changed from MER to two upcoming Mars missions. We'd be happy to tighten/revise/technicalize it significantly if it would be of interest to folks at IWPSS.

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. Experience designing, building and operating the MER tools with our colleagues, studying the use of the MER tools from a Human/Computer Interaction perspective, and feedback from these three missions has lead us to take a somewhat different approach to designing and deploying applications with planning technology this time around. 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.

Introduction

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) (Norris J. S., et al, 2005b), a tool used to create constraints between activities (e.g., the rock grinding must take place before the picture taking activity that documents it), 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 (Bresina, Jónsson, Morris and Rajan, 2005a). 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 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 Phoenix, 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:

- 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
- 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

- 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. In this case, Europa computes new times for activities or may unscheduled activities that introduce violations that cannot otherwise be repaired. The changes Europa introduces can be undone, and each activity that is moved is visually distinguished and displays the change Europa has made.

The main point is that the planner is always acting in either a passive, informational way or acting act the direct request of the user in a way that can be easily undone. 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.

This change in the way power that the user leverages the power provided by automated planning technology provides required some changes to the way the domain is modeled, to the planning algorithms, and to how the planning problem is set up and maintained. Bresina and Morris describe these changes in detail in their 2006 SMC-IT paper (Bresina and Morris, 2006).

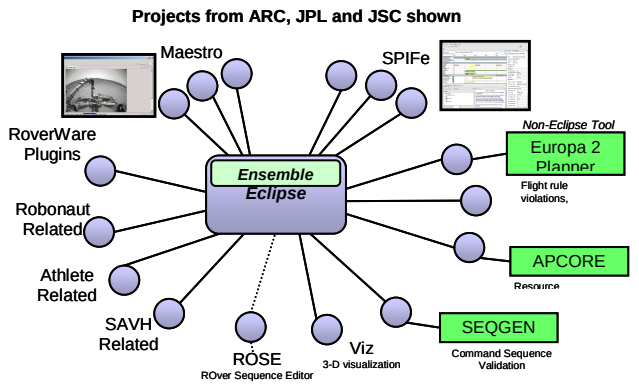


Figure 1: Plug-ins and services developed for Ensemble

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. These tools were not initially designed to be part of same conceptual process and contained many duplicated, inconsistent interfaces to same subtask. In addition, the plan moved between tools via translations that were most often one-way. Thus once a plan had been moved from one tool to the next, modifications that required functions in a previous tool could no longer be easily made.

Slightly less obvious is the problems this causes during development. Because the tools are not designed to be used together, they are typically not integrated until fairly let in the development process, and in a somewhat ad-hoc manner. Since integration is done via scripts and translations, it's not always clear when the integration is broken, as opposed to some more programmatic, type-checked integration style. In addition, a significant amount of time is wasted as each team responsible for a tool develops incompatible and redundant data formats, displays etc. Ironically, more time is wasted later integrating them all back together.

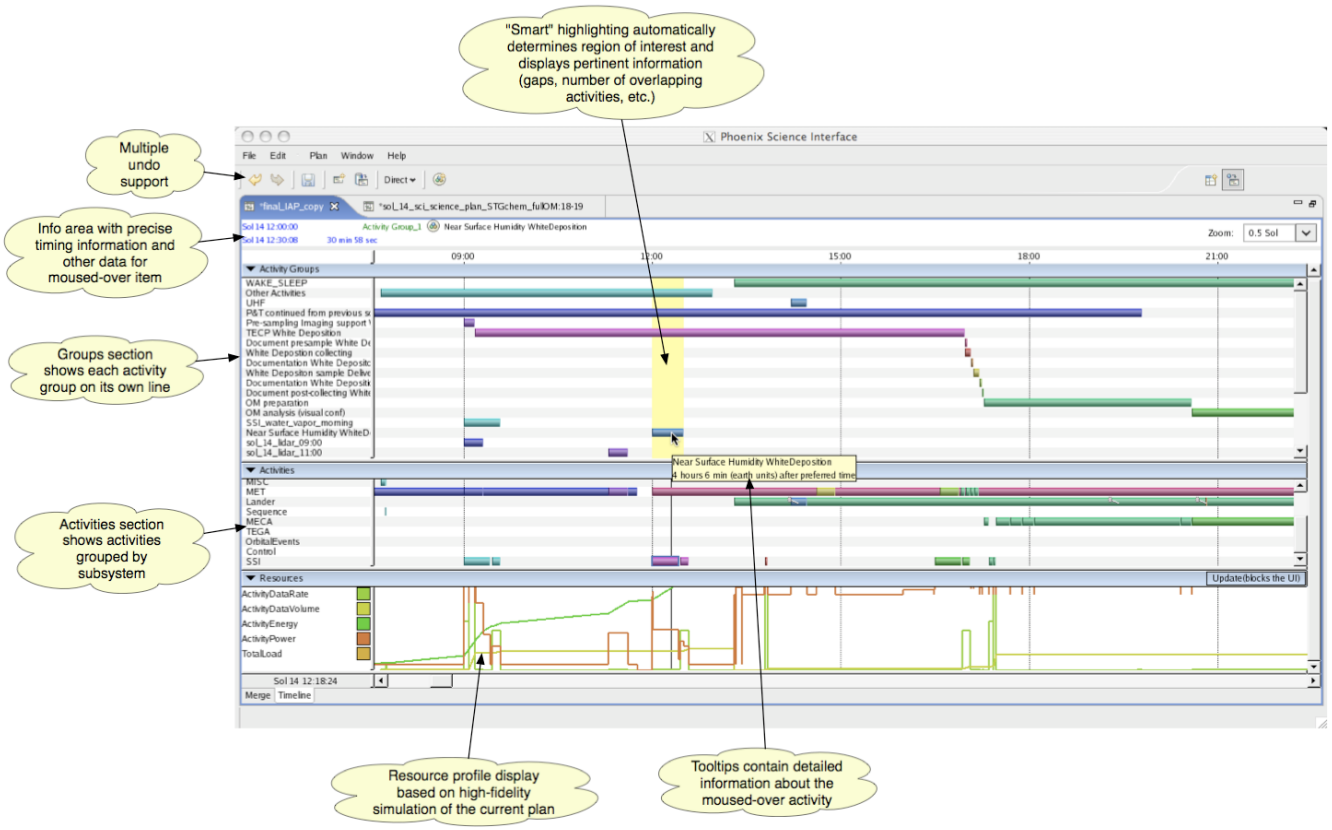
Ensemble is a software platform for operations, inspired by the MER experience, which allows software from different teams to be integrated together into a single ops application. Ensemble is based on components, meaning different software functions are delivered as re-usable components that can be combined 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. Figure 1 gives a flavor of the number and variety of teams involved to date and the names of their contributions to Ensemble. A variety of software components or plug-ins for different tasks —from manipulating scientific images to validating complex sets of spacecraft activities — have been plugged into Ensemble. The figure also indicates that the Ensemble platform is heavily based upon Eclipse, an open-source, component-based Java software platform, which we discuss briefly in the next section.

The core parts of Ensemble, indicated schematically in the box labeled Ensemble, contain functionality that can be shared by all Ensemble plugins. This includes shared data structures to represent plans and lower level command sequences. These shared data structures take the place of file and script integration between components, and can be checked at compile time as each component is being implemented or modified. Ensemble also contains shared implementation of routine tasks such as I/O to common formats used by external mission tools, shared implementations of simple activity viewers or other components that have been designed or implemented collaboratively, and so on. On top of this layer of functionality, specialized teams integrate their higher-value Ensemble plug-ins.

In addition to the Java plug-ins that are directly implemented with Eclipse, Ensemble 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.

Europa 2 (**need a reference**) is a planning system developed at NASA Ames Research Center which we have adapted and wrapped as such a network service, with corresponding Eclipse plug-ins that allow information to be exchanged between the planner and other Ensemble plug-ins. The Europa 2 server is a multipurpose planning platform core with an XML-RPC server layer above it used to interface with the Ensemble client. The server API is designed to use the underlying planning platform as a mixed mode planning advisor for the client.

Figure 3 is SPIFe, a set of Ensemble plug-ins designed and developed by the authors based upon experience with the MER operations process. SPIFe includes many several views for laying out plans on a timeline, editing



constraints, and receiving lists of plan flaws and explanations from a planning system such as Europa 2. Since SPIFe itself is comprised of a number of plug-ins, various subsets of its features may be deployed in different applications.

Not shown but important to the planning process is another set up plug-ins collectively referred to as Maestro. From a purely planning-centric perspective, the Maestro plug-ins allow scientists to define what activities shall be planned and connect them with real-world objects in the rover or lander's reference frame. These plug-ins developed at JPL allow scientists to browse science images and other data delivered from a spacecraft, create targets in a 3-dimensional reference frame based on stereo images, define activities on those targets, and make initial estimates of their resource usage before final planning. The Maestro plug-ins and services provide MER extended ops, the Phoenix mission and MSL with the initial activity planning process and the ability to operate with scientists distributed around the globe on. Unfortunately the details are largely outside the scope of this paper, but can be found in (Norris, et. al, 2005b) and (Fox et al. 2006).

It's important to note that in addition to providing software plug-ins, Ensemble is also a software development process by which many teams from different centers can cooperate to develop mission operations software. It includes shared code and documentation repositories, early integration of compatible components, and continuous automated building and unit test of integrated products.

Use of Eclipse

The open-source Eclipse platform is an essential tool for development within Ensemble. Eclipse provides the basic structure for plug-in based development and management of multiple products based on selections of plug-ins. It allows us to assemble applications from chunks of functionality developed by different teams in a surprisingly fluid fashion because cooperation is no longer dependent upon external interfaces (file IO, network communications, etc), but can instead implement application programming interfaces (APIs) that are flexible, robust, and checked by compilation any time code is modified.

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. This leads to Eclipse's plug-in architecture that provides Ensemble with the flexibility necessary to mix and match features for particular missions.

One visually dramatic example of this mechanism is 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 defined by Maestro to the planning and scheduling perspective defined by SPIFe with the click of a button. This functionality essentially melds the functionality of two entirely different applications into a single desktop executable.

Some additional benefits of extending Eclipse for use in Ensemble applications are that it provides API's to handle window management, file IO, dialog creation, developers can focus upon their core competencies, it provides a consistent look and feel and metaphors across the entire application, and it provide cross platform compatibility.

Simple, agnostic interfaces

In the Remote Agent architecture flown as experiment on the Deep Space 1 spacecraft (**Needs a reference**), the planning system had hooks which would call out to services supplied by the spacecraft flight software. These services would perform specialized calculations such as determining how long an engine burn should last given some parameters. In some sense, in integrating Europa 2 with Ensemble, we have turned this model inside out. Europa 2 can be seen as a set of services that can be called by mission applications through a simple API to answer queries about activities, constraints between them, how to modify them, and so on.

We believe simply turning how we think about the planner inside out has provided a significant amount of traction. One way to lower the barrier to adoption is to decrease the complexity of implementation. The fact that the planner is a self-contained service that a mission application simply calls, rather than having major system integration tasks within the planner, means it is much easier to add the planner or remove it later if necessary. Ironically, this means it is much easier to get the planner approved in the first place.

As it is unknown what platform, language, or architecture will be used by any particular mission system, it is necessary to devise a method by which most any system can interface with our planning system. 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 core Ensemble applications written in Java and external services such as Europa 2. 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 to adding, modifying and deleting activities and constraints and modifying activity parameters. For our customers, we distinguish between temporal constraints between activities that are added via the API, and constraints inherent to the Europa 2 model, which we refer to as flight rules. 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.

An interesting side effect of this style of interaction between the application being deployed in Ensemble and the planning services provided by Europa 2 is that applications that have nothing to do with Ensemble can easily be adapted to connect to Europa 2 via its API. In addition, Ensemble plug-ins that interact with the planner, for example to display flight rule violations or highlight activities that are in violation, could be connected to any planner that can support this API. As a result, we've started a project to connect Europa 2 to an existing web application based on this API, and developers of another planner are investigating use of Ensemble with their planner.

Models generated from existing mission artifacts

Europa 2 has a sophisticated and general language for describing planning domains and planning instances called NDDL (**Needs a reference**). Our intent for mission deployments is not to use it. In the case of MER, the planning tool MAPGEN used Europa and a planner language called PDDL. MER mission personnel created an XML document called the Activity Dictionary (AD) which described the activities the MER rover could perform, and a planning expert created a PDDL domain that captured these activities and their interactions. This resulted in a domain encoding that was as efficient as possible, but was often out of synch with the AD or the myriad other tools whose authors created domain models in a similar fashion.

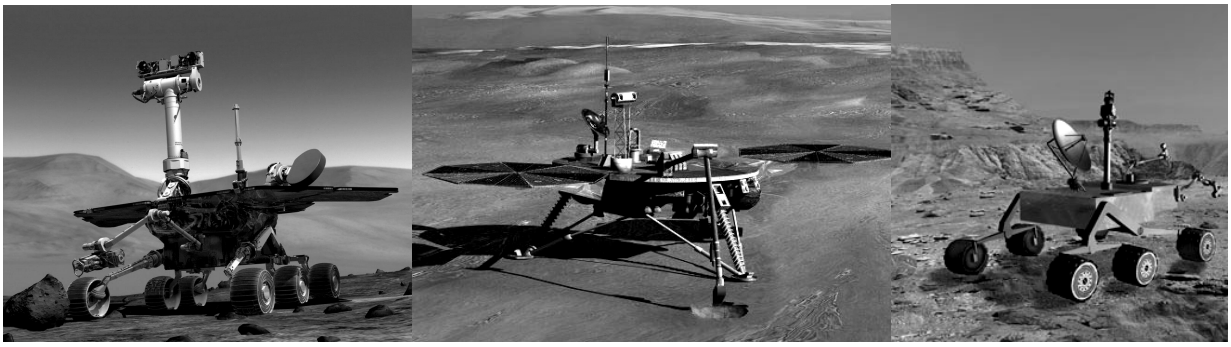


Figure 3: MER rover, Phoenix lander, and MSL rover (not to scale)

For the use of Europa 2 in Ensemble, we have two strategies for avoiding the inevitable disconnects between a mission's AD and our domain description in NDDL. The first strategy is not to have a domain description. If every activity by a user has only a name, start and end types, and temporal constraints with other activities, it is still possible for Europa 2 to find temporal constraint violations, provide bounds for constrained move, and suggest strategies for resolving conflicts. These are very useful services for many applications, and given the existence of the Europa 2 server and plug-ins to use it from Ensemble, can be deployed almost for free. This is the strategy deployed on the Phoenix mission.

The second strategy is to automatically generate a NDDL description of the planning domain from a mission's AD. NDDL is a very general planning language, but an analysis of the MER model used with Europa revealed that very few idioms were being used to represent the operating restrictions on the rovers. These typically involved specifying which resources (e.g. sole control of the rover's arm) each activity required, what state condition an activity required or set (e.g. the deploy operation leaves the rover's arm out). On MER these couldn't be generated from the AD because of course no one thought of including the needed information in the AD a priori, and they would result in domain model that was less efficient than the hand written model in terms of the number of objects and predicates used to describe the rover. We approached the first problem by designing a very limited language of activities and resources that could be used to generate the NDDL model. We tested it by generating a MER NDDL model from a simple specification and running Europa 2 with it on archived plans from MER operations. Emboldened by that success, we then worked with MSL to incorporate the necessary information for NDDL generation into the schema for the AD that will describe their mission. We anticipate generating domains description for MSL from their AD, keeping Europa 2 and other mission tools in synch and enabling us to continuously deliver prototypes as the set of activities in the AD is expanded and modified during the development of the mission.

As far as the difference in efficiency between a hand written domain model and the generated model, the naively generated model to be sure has redundancies that are causing the planner to do more work than is necessary. However, between changes to the way we are invoking the Europa 2 through its narrow interface and efficiency improvements between Europa and Europa 2, the Europa 2 response on the generated model is significantly faster than Europa on the hand-made model. We had considered performing a second pass through the generate model to remove the most obvious redundancies, but it has not yet seemed to be a high priority use of development resources.

The Missions

Before discussing the advantages we feel this approach may offer, we provide a little more detail on the three Mars missions where we are currently deploying Ensemble-based applications, in addition to various research uses of the system. Figure 3 illustrates the spacecraft for these missions, but does not faithfully represent their relative sizes.

MER

Operation of the two MER rovers, Spirit and Opportunity, has continued far past their designed ninety day primary missions. Our Ensemble partners at JPL have delivered an application consisting of their Maestro plug-ins to MER extended operations, where it is used daily to browse images downlinked from the rovers, create targets, define activities, and make initial estimates of their resource usage. Currently the output of the new Maestro tool feeds into the same activity planning process used in the MER primary mission for final planning. It's worth noting that the overall process design for how users would interact with Maestro was developed with the HCI experts at NASA Ames, both to improve the user experience for MER and to ensure that Maestro plug-ins be consistent with other Ensemble plug-ins when deployed in larger applications.

Phoenix

The Phoenix Mars lander launches 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.

The Ensemble tool that has been delivered to the Phoenix mission is called the Phoenix Science Interface, or PSI. The PSI application consists of the components delivered to MER for Maestro, plus the SPIFe components for developing activity plans, the Europa 2 server, and APCore, a JPL system for validating plans, the functionality of which overlaps with Europa 2.

Interestingly, Europa 2 was not originally baselined for inclusion in PSI due to the tight budget and schedule of the Phoenix, perceived risk in using automated planning, and the overlap with APCore. As we developed the Europa 2 server and associated Ensemble plugins for MSL, it became natural to demonstrate it and eventually deploy it for Phoenix. Since APCore handles many plan validation tasks for Phoenix, Europa 2 is used largely for temporal constraint validation and repair, which unlike flight rule validation can be done without developing a Europa 2 model of the spacecraft. Given the existence of the generic Europa 2 server, this was very low cost and low risk to the mission.

MSL

The Mars Science Lander is a large rover scheduled for launch in late 2009. Its landing site has not yet been selected as scientists digest the large amounts of information being returned by the MER rovers and orbiters such as the Mars Reconnaissance Orbiter. Based on lessons learned from previous rover missions and a number of new technologies ranging from how the vehicle lands on Mars to a laser to remotely interact with rock samples, MSL will deliver a significantly larger array of scientific instruments to Mars, will be able to traverse more difficult terrain, and give scientists far greater latitude about choosing where on the planet land.

Because of the larger number and greater complexity of scientific instruments on MSL as compared to MER and Phoenix, we anticipate MSL will have the greatest need for appropriately-focused planning technology to help scientists and engineers develop valid, high-throughput activity plans and command sequences before they must be uploaded to the spacecraft. Fortunately, MSL also has the longest lead time, with Mars surface operations beginning in 2010. Due to our use of re-usable but extensible components and a planning service driven by the mission's own language for describing activity types, we have been able to easily generate a prototype MSL activity planning system which we will deliver in late 2006. The MSL system contains most of the functionality of the PSI system delivered to Phoenix, plus significantly more use of Europa 2 for detection and repair of flight rule violations, computation of when the spacecraft should be active versus shut down to save power, and so on. As this prototype evolves into the MSL activity planning and sequencing system, we anticipate it will absorb the lessons learned from developing and operating the Phoenix mission, as well as requiring and revealing new ways for planning technology to speed up the tactical operations process.

Advantages of the Approach

A great advantage of turning our planner into a planning service with a narrow API is the flexibility it gives us as to how we fit into an operations process. An often-heard issue with the MER activity planning and scheduling system was that it is set up as a string of pearls – each distinct tool takes the input from another tool, translates it, processes it, and writes a file that is to be read by the next tool in the process. Thus the order and method by which each tool is used is cemented into the set of file transfers and translations that glue the tools together. In contrast, all Ensemble plug-ins in an application such as PSI are operating on the same data structures, and can be used in any order, and external services, such as Europa 2, are accessed through a narrow API that can be called by an application whenever the information required is available. What this means is that information that Europa 2 can provide, such as the earliest possible time an activity can start given its constraints, or what flight rules are violated, ends up being used in ways we never initially imagined. For example, in PSI there a "to do list" where scientists can collect sets of activities they'd like to propose for the lander. After implementing it, it became obvious that showing the earliest possible start time for each activity that Europa 2 can compute given the other activities and constraints would be a helpful but unanticipated addition.

This touches on the advantage of incremental deployment. Having planning applications that are assembled from components and a planner that provides several sets of services via simple APIs makes it significantly easier to add planning technology to existing applications. Applications based on Ensemble can be deployed with or without the plug-ins that drive the connection to the Europa 2 server. Since adding them is relatively light weight, that decision need not be made irrevocably before beginning an application. This makes it significantly easier to prototype the inclusion of planning technology into an application when the opportunity naturally arises, test it with users, and demonstrate it to mission stakeholders. This lowers risk and increases customer confidence and opportunity for feedback, as was the case with PSI and may be done with MER extended operations. In addition, the API we developed to enable Ensemble plug-ins to drive Europa 2 can be called by software that

has nothing to do with Ensemble. Thus we've been approached by developers of web-based and other applications to provide low-cost access to planning capabilities.

Another major advantage 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 risk, and once people are familiar with it there is a major advantage to additional missions using it both in terms of training and trust. Low cost incremental improvements can be applied to software to benefit from lessons learned on each mission while still leaving the core software. As mentioned, the majority of the Ensemble code delivered to the MER mission as Maestro has also been delivered to the Phoenix mission as a part of PSI, with the addition of mission specific plug-ins to account for the different instrument packages. To these Maestro plug-ins we'll add components for SPIFe, for connection to Europa and APCore, and so on to form the complete PSI. Similarly, most of PSI will be delivered to MSL, with additional plug-ins to specialize it for the MSL rover and to increase its capabilities. Ironically, some early MSL development work, such as that related to Europa 2, has been delivered to Phoenix, and when MSL flies will be "Phoenix heritage" components. In addition to building re-usable plug-ins and services, we participate directly in the system engineering of the activity planning systems for the missions we are delivering to. In doing so, we serve as one more conduit for mission best practices and lessons learned to flow between missions and into our products.

Finally, in addition to having an HCI team as partners in driving development of these planning applications, they perform structured user testing of the applications with the scientists and engineers who will be the users of the system and help to define at least quarterly demonstrations and deliveries to the customers. We believe this greatly reduces the likelihood that we will be focused on the wrong capabilities or the wrong problem for very long.

Risks That Must Be Managed

We are enjoying a great savings by deploying the same planner service and largely the same planning application across missions, and by sharing the implementation of routine functions across teams. However, there are risks that come with this style of development.

Since we are serving multiple missions, we could find ourselves in a position where conflicting mission requirements make it difficult to define what the functionality of our application should be, or force us to implement a least common denominator solution. Thus far, we've found that the combination of building applications from plug-ins and making use of Eclipse's extension point mechanism to customize functionality of plug-ins for each mission where necessary have allowed us to avoid this problem.

Since our planning server and Eclipse plug-ins are used in multiple products, each devoted to a particular mission or research task, there's a real risk that changes made to a plug-in to satisfy one customer will inadvertently break the application being delivered to another customer. Our most direct line of defense against this risk is automated testing. Whenever a change is made to a plug-in, all products that use the plug-in are automatically rebuilt and run through a series of unit tests. A second line of defense is our practice of keeping the core planning server and our Eclipse plug-ins mission independent. The planning server is extended with different spacecraft models or specialized code for computing mission specific items such as heating profiles, outside of the main server code. The Eclipse plug-ins are refactored into a mission-independent portion and a mission-specific extension point when it's found that a mission-specific solution is needed. In this manner, code that will only work for one mission is explicitly tracked as part of the deployment for that mission only. Finally, by constraining our planning server to focus on a narrow API that captures what we believe it is best at and leaving everything else to be computed within a client program, we believe we've reduced the risk that some feature we develop or computation we provide will be inadvertently adjusted in a way that's appropriate to one mission and not another.

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 importance of easy 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.

Acknowledgements

The authors would like to acknowledge the developers of the Science Activity Planner, Constraint Editor and MAPGEN tools delivered to MER, without which the Ensemble collaboration would not have begun. They would also like to thank all of the Ensemble development teams, especially the Maestro team led by Jeff Norris at JPL. In addition we are grateful to the MER, Phoenix and MSL missions for the opportunity to participate in this adventure.

References

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

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

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

Fox J. M., Norris J. S., Powell M. W., Rabe K. J., Shams, K. S. 2006. "Advances in Distributed Operations and Mission Activity Planning for Mars Surface Exploration", *AIAA SpaceOps 2006*, Rome, June 19-23, 2006.

Norris J. S., Powell M. W., Fox J. M., Rabe K. J., Shu I. 2005a. "Science Operations Interfaces for Mars Surface Exploration," *2005 IEEE Conference on Systems, Man, and Cybernetics*, Big Island, HI., October 15, 2005.

Norris J. S., Powell M. W., Vona M. A., Backes, P. G., Wick, J. V. 2005b. "Mars Exploration Rover Operations with the Science Activity Planner," *IEEE International Conference on Robotics and Automation*, April 2005.