Science Operations Interfaces for Mars Surface Exploration

Jeffrey S. Norris, Mark W. Powell, Jason M. Fox, Kenneth J. Rabe, I-Hsiang Shu
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109
Jeff.Norris, Mark.Powell, Jason.Fox, Kenneth.Rabe, I-Hsiang.Shu@jpl.nasa.gov

Abstract – *The Science Activity Planner (SAP) is the science planning tool used for the Mars Exploration Rover (MER) mission. This paper begins with an overview of the software developed for MER and how it was used for science downlink analysis and activity planning. The overview of SAP is then followed by a report on a number of new development efforts that aim to improve on the capabilities of SAP for future missions. The selected areas discussed herein include the application of geographical information systems in tactical downlink analysis, new strategies for distributed data access and planning support, a virtual field test capability to support "what if?" scenarios for new technologies and mission concepts, and the use of agile development methods to improve the development of Mars surface mission support software as a whole.*

Keywords: Science activity planning, spatial databases, distributed operations, virtual field testing, agile development

1 The Science Activity Planner

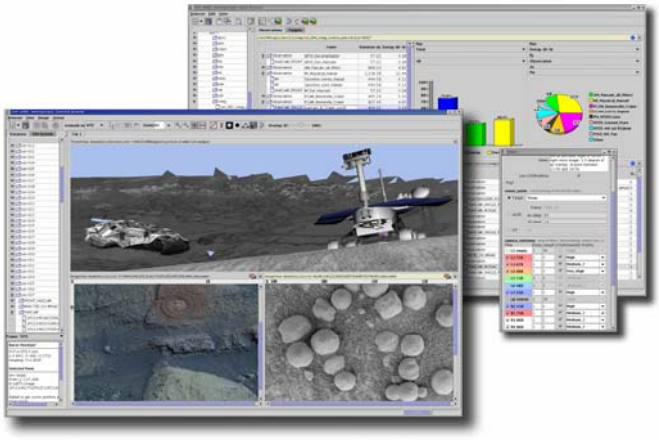

Figure 1. The Science Activity Planner with Downlink Browser in foreground, Uplink Browser in background [1].

The science planning tool for the Mars Exploration Rover Mission is the Science Activity Planner (SAP), shown in Figure 1. SAP assists the Athena Science Team and their collaborators in two main areas: downlink data analysis and science activity specification. Downlink analysis begins with viewing images and other data products, followed by the interactive specification of new science targets and activities. The image views provide a contextual awareness for the scientist where they interactively specify new activities for the spacecraft and its instruments.

1.1 Downlink Analysis

On each sol (Martian day) of operations, data arrives from the rover and is processed by the Multi-mission Image Processing Lab (MIPL) pipeline to produce data product files. These products include images and spectra along with metadata describing the state of the rover when the products were acquired. A SAP user chooses the products he/she wishes to view and arranges them as panes in a reconfigurable data visualization area called the Viewgrid (shown in the foreground of Figure 1.). Users can configure the number and relative sizes of Viewgrid panes in order to suit their needs, or add additional pages to the Viewgrid if more space is needed. The Viewgrid enables the user to display numerous data products simultaneously without the visual clutter that would result from a multiple window system.

Images acquired by the rover can be viewed in a variety of ways in SAP as shown in Figure 2. Individual images from the rover's mast cameras can be combined into a collection that SAP will warp together on the fly to produce a panorama image. These panoramas can also be viewed in an overhead polar projection or an immersive 3D view that allows a user to position a virtual camera within the scene.

SAP provides basic image viewing features such as zooming and panning and image processing features such as high pass, low pass, and edge detection filters. SAP also provides a set of image analysis features that are tailored the particular needs of the mission science team. For instance, scientists use the color information returned by the Pancam instrument to assess the expected relative abundances of important minerals. SAP users can perform this kind of analysis by creating synthetic color images using arithmetic functions based on the available bands of an image. SAP also includes a hyperspectral visualization tool called the ImageCube that allows a user to analyze the rich dataset returned from the Miniature Thermal Emission Spectrometer (Mini-TES) instrument.

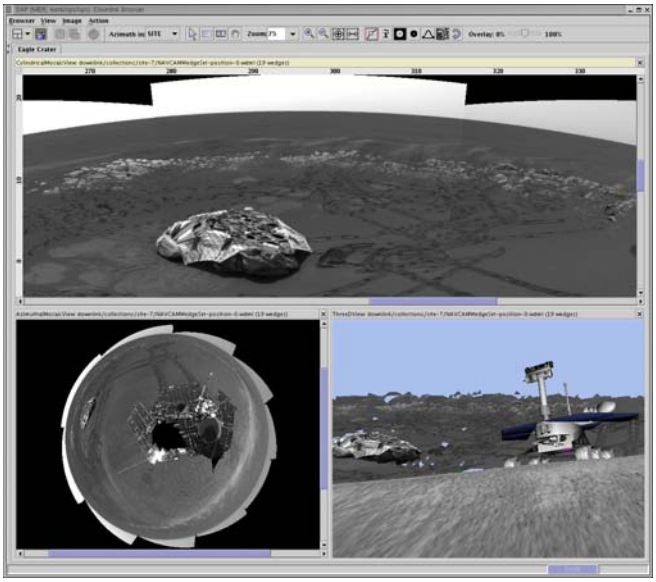

Figure 2. Cylindrical (top), polar-azimuthal (bottom-left), and 3D (bottom-right) renderings of a mosaic of images acquired by Opportunity [1].

When a SAP user selects an image to view, SAP also loads range data derived through stereo correlation. When a user clicks on a point in the image, this data is accessed and the user is presented with the position of that point relative to the rover in Cartesian and spherical coordinates. This range data can also be overlaid on the image as a color-coded elevation or range map. Images taken of the area immediately in front of the rover with the front Hazard Avoidance Cameras (Hazcams) are also accompanied by reachability maps indicating where the rover's arm can be safely deployed. SAP enables its users to overlay this information to allow them to select suitable locations for in-situ measurements.

1.2 Activity Plan Creation

SAP users transition from analyzing data received from the rover to planning new rover actions by using SAP to mark locations of interest. There are two types of these markers. Features are used to indicate objects of interest such as a boulder, a patch of sand, or a distant hill. Targets are used to indicate a specific location within a Feature and can be used as parameters in an activity. For instance, a scientist might create a Feature to name a rock "Pumpkin" and then create a Target called "Stem" near the top. Features and Targets are rendered in all SAP views and are immediately shared with all other SAP users to ensure that scientists don't create Features or Targets with conflicting names.

After creating Features and Targets, a scientist switches to the SAP Uplink Browser (shown in the background of Figure 1) in order to create Activities for the rover to accomplish. Each Activity added to a plan is drawn from a predetermined set of Activity types. For example, one Activity type is used to drive the rover while another is used to take a picture with the rover's navigation cameras. Once an Activity has been added to the plan, the user can customize its execution by adjusting the values of its parameters. The Activity used to request use of the Microscopic Imager instrument allows the user to indicate the point that should be imaged (typically a previously created Target), desired compression ratios, the priority of the data that will be acquired, and so forth.

Activity plans for Spirit and Opportunity are constructed in a series of meetings. On each sol, SAP is initially used by individual users who construct fragments of plans as Activities and Observations. These users meet in small Science Theme Groups that are responsible for integrating these fragments into a plan that reflects their particular scientific interests. Next, representatives of each science theme group meet in a large meeting called the Science Operations Working Group. This meeting is responsible for integrating the plans from each theme group into a single, authoritative plan for the next sol. This plan is refined further before it is transmitted to the spacecraft. At each step in this process, less suitable Observations and Activities are discarded and the surviving elements are prioritized to reflect their relative importance to the operations team.

SAP is designed to support this process of iterative integration and prioritization. The Uplink Browser contains a ViewGrid that is similar to the ViewGrid that exists for viewing images that was described above. By subdividing the Uplink Browser ViewGrid, a user can load several plans simultaneously and merge them by dragging and dropping Activities between them. This also allows users to refer to previous plans and reuse portions of them in new plans.

1.3 Simulation

SAP also enables a scientist to simulate the expected effects of an activity plan. Critical feedback includes the amount of power, data volume, and time the plan will require, as well as the expected final position of the rover and the instrument arm. SAP also draws an "image footprint" on top of all image displays to indicate the region that is expected to be imaged by an Activity. All of this feedback enables a user to adjust the plan until it reflects his/her intent.

SAP employs an efficient approach to plan simulation in order to provide this feedback to the user in a timely manner. As the user constructs a plan, SAP maintains a sophisticated dependency graph that keeps track of exactly which Activity parameters will impact particular spacecraft states. This dependency graph allows SAP to determine exactly which parts of a plan need to be simulated when a

parameter is modified. In practice, only a small portion of the plan is affected by any particular parameter change, so this dependency-based approach to resource modeling provides SAP a tremendous performance boost. SAP is able to simulate the effects of most changes instantaneously, providing the user immediate feedback and improving their planning efficiency immensely.

1.4 Building on SAP: Maestro

SAP was successful as a science analysis and planning tool. After its initial deployment in support of the MER nominal mission, we dubbed the next version of the software "Maestro". We are now adding new capabilities to Maestro to further increase the efficiency of scientists for MER and future rover missions. The areas we are currently focusing on are geographic information systems, distributed operations, virtual field tests, and agile development. We will explore each of these areas in further detail in the following sections.

2 Geographic Information Systems

A Mars rover mission such as MER collects images at an ever-increasing number of different sites on the planetary surface. The current state of the art in data browsing tools used in rover operations presents an image catalog in one of two organizational strategies. The first strategy is a temporally-ordered image catalog, where images are organized in a hierarchy based on time of acquisition. SAP and most current operations tools fall into this category. The other type of organization that has been used in operations is round-trip data tracking. This type of tracking produces a correlation between each image product and the spacecraft activity that was executed to acquire the image. Based on this correlation, an activity planner can review which activities have results received in full, in part, or not at all.

2.1 Spatial Indexing

Spatial data indexing applied to Mars rover images and science data products adds a useful new cataloging modality for tactical data analysis. Spatial indexing of image products can serve to visually indicate where on the surface the products were acquired. An orbital image that covers a significant area of the rover's traverse path can be annotated with the locations where surface images were captured. Queries for images can be expressed in terms of the location on the surface where they were acquired, using both the rover's traverse path and the contextual surface features that are visible in the orbital image. The combination of these capabilities results in a map-like user interface that is highly intuitive (based on a person's prior experience using maps), especially to the scientist-activity-planner whose training is in geology. To the geologist, using a map to place in situ observations and other results into a larger regional context is a fundamental tool of field work, and field geologists represent a large segment of the user community of operations interfaces for planetary rovers.

For each data product acquired, a Mars rover records its position and attitude at the time of acquisition. This information is part of the image metadata that is returned with the image in the downlink. As the rover moves through different locations on the surface, this set of locations is recorded and converted to a chain of coordinate frames, where each new frame is defined relative to the previous frame. The rover's current position relative to the landing position at the start of the mission is obtained by concatenating all of the coordinate transformations back to the landing location. However, we would like to have a precise location of all of these rover positions in absolute coordinates on the surface, not merely lander-relative coordinates. New high-resolution orbital cameras such Mars Express' High Resolution Camera (HRC), and new techniques for acquiring high resolution imagery using older orbital cameras such as the Image Motion Compensation demonstrated recently with the Mars Global Surveyor's Mars Orbital Camera (MOC) can accurately locate landers such as the Mars Exploration Rover's landing sites at Gusev Crater and Meridiani Planum. Such high-resolution images provide an absolute geospatial reference for the lander position, and all of the relative coordinate frames along the rover's traverse path can likewise be determined in absolute coordinates by concatenating the frame information. In this way we compute the 3D and latitude-longitude location of every rover position on the surface.

2.2 Spatial Databases

A spatial database is the foundation of a Geographical Information Server or GIS. With a GIS database, SQL queries are used to insert, update, and select entries in a relational database that can contain geometry information in addition to standard types of data such as text and numeric information. The types of geometry that a GIS database can support may vary from a simple 2 or 3 dimensional point, to connected line segments or polygons. For our implementation, we use several of types of geometry: 3D points to represent locations where images were acquired in x,y,z space, 2D points in latitude and longitude coordinates to represent where on the planetary sphere images were acquired, and a polyline to represent a set of connected line segments with 3D endpoints that represents the path of the rover's traverse.

For our implementation we chose to use the PostGIS spatial database infrastructure. The PostGIS spatial database has several advantages for supporting tactical operations. It is freely-available, and has long been proven capable of industrial application support. It is also open source, which makes customization of the framework and

detailed understanding of the implementation straightforward.

2.3 User Interface

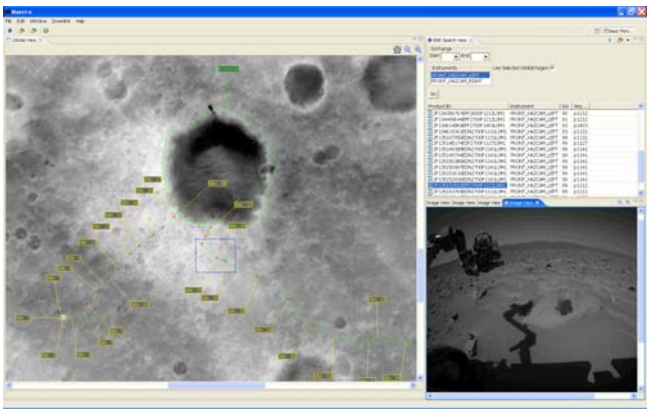

Figure 3. The Maestro Orbital View. A MOC image of Gusev Crater, Mars is shown, annotated with landmark and rover traverse locations provided by GIS.

The GIS user interface (see Figure 3) renders an orbital image using a simple cylindrical map projection. The image can be resized to any arbitrary scale using zoom in/out controls. The image is annotated with a number of useful landmarks. Users can outline and name new landmarks of interest using a polygon drawing tool that stores the landmark positions in a database. For supporting rover missions, the rover traverse path is overlaid onto the image as a connected series of line segments connecting all of the waypoints along its traverse history. The rover's traverse path overlay is also labeled to indicate the numeric designated of the major sites along the path. The labeling uses a greedy algorithm that places as many labels as can fit on the image to annotate the path while still remaining legible, depending on the current zoom level of the image. For image product queries, the user can click and drag a rectangular region on the map to search for image products that were acquired within that region on the surface. The query is executed against the GIS database and the resulting set of data products are presented to the user in a result table. The GIS queries are highly efficient since the PostGIS implementation maintains an R-tree data structure to optimize spatial queries. As a data product is selected from the query result set, its corresponding map location is highlighted with an icon.

3 Distributed Operations

Centralized operations worked initially for MER because everyone was co-located in a single room for planning. However, as the mission progressed beyond its expected operational lifetime, support for distributed operations was required as scientists returned to their home institutions. This problem was addressed with development of Remote SAP [2].

Knowing that future Mars missions will likely have operations lifetimes that will exceed the projected lifetimes, planning for distributed operations is prudent. The two features that are needed for distributed operations for future missions are distributed planning and distributed access to downlink data. By planning for distributed operations at the beginning, we hope to avoid some of the pitfalls and problems that were encountered during MER.

3.1 Distributed Planning

In SAP, plans were stored on the local filesystem, which is a non-ideal situation when working in a distributed operations environment. Remote SAP still uses local files to store plans, but has a manual procedure for remote users to contribute to the master plan. While this works for MER, this begs a more effective solution for missions with longer operational lifetimes.

In our recent efforts, we are now using a relational database as the central point of contact for the storage and distribution of plans. This has the advantages of maintaining a single authoritative plan data store and supports collaboration, allowing all contributors to submit their pieces of a daily plan in an automated fashion. Using a database also allows for more robust querying of previous plans as queries can be built using any data available in the database. This is an improvement over plan querying in SAP, where the only available mechanism to find any plan was through a rigid filesystem directory hierarchy.

We are using a technology called Hibernate to handle the majority of the database interactions with a Postgres relational database. We have been using Hibernate to define the database schema and generate code for all or our core model objects. We significantly enhanced the code generation over the stock Hibernate code generation to better support both Java 1.5 and common capabilities that we have identified as being necessary for our development.

3.2 Remote Data Access

For MER data was pushed out to the users by the File Exchange Interface (FEI) data file subscription service. Unfortunately, this method required significant investments of storage for data that may be accessed at best infrequently. To reduce the amount of data pushed out to users, the Maestro team is adopting a plan to treat downlink data more similarly to the web model.

All downlink data products will be available through a server. When a client attempts to access a data product, the data product will be downloaded to the user's machine and stored locally. Having the data stored locally gives

significantly improved access times for the user. Additionally, this model allows the ability to bring a new collaborator into the project quickly without requiring the user to have all the mission data.

Storing the data on the user's local disk carries with it the logistical issues of synchronization, disk space limitations, and local file system organization. Fortunately, the first two problems are similar to the problems that have been solved by web browsers. File creation/modification timestamps can be checked to determine whether a file is stale or not. Having a separate file that tracks some meta-data about downloaded files will allow us to manage the cache so that the user does not need to worry about running out of disk space. To handle the problem of locating the file, the filesystem directory structure that will be on the user's local machine will mirror that of the server, which allows for a quick check for file existence to determine whether the file needs to be downloaded or not.

4 Virtual Field Tests

4.1 The Costs Behind Field Tests

Prior to launch of a Mars surface mission, a series of readiness tests must be performed in order to demonstrate the operability of the technologies used. In addition, science and engineering teams need to be trained in the tools and technologies that will be supporting the mission. Realistic field tests are conducted to address these needs involving a team of engineers who deploy a research-class or flight-prototype rover to an area similar to Martian terrain (e.g. Mojave Desert) to support remote science operations.

Performing field tests significantly increases the cost of a mission primarily because it requires a large effort to transport and sustain the necessary team and equipment at the test site. Additionally, these tests need to be conducted many times for months or years as each test generates new results, and as new technologies and operations techniques are tested and validated. More training opportunities are also needed as the size of the science team for a mission grows incrementally over the course of years.

4.2 The "Field Test in a Box"

Virtual field tests provide a way to dramatically reduce the costs of these actual field tests to a particular mission. Offering a "Field Test in a Box" experience that can be setup at any facility, the virtual field test can be used as a low-cost way to educate and train scientists and engineers or to demonstrate some aspects of operational readiness. It does not carry the large financial and logistical overhead of an actual field test. Of course, virtual field tests serve only as a complement to actual field tests since the need for field tests with actual hardware in a real environment is inescapable. However, the virtual test will provide a "jump-start" for scientists and engineers, enabling them to reduce the number of iterations out in the field both during training or technology demonstrations.

4.3 RoverWare Architecture

RoverWare is an end-to-end virtual field test system that represents the integration of three existing JPL technologies:

1. Maestro – The scientist's user interface for downlink analysis and science plan specification.

2. CLARAty (Coupled Layer Architecture for Robotic Autonomy) – a reusable software architecture that provides an extensive library of robotic functionality that simplifies the integration of new technologies onto robotic platforms [3].

3. ROAMS (Rover Analysis, Modeling and Simulation – a physics-based simulation tool for analysis, design, development, test, and operation of rover on planetary surface exploration missions [4].

These three technologies are integrated as shown in Figure 4.

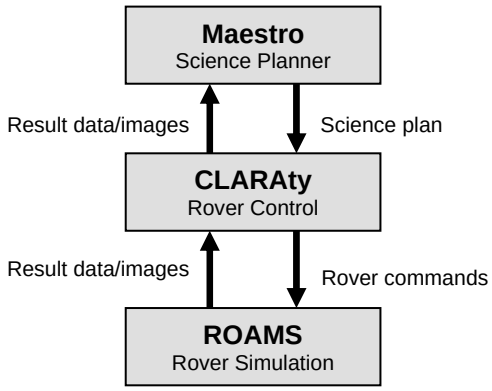

Figure 4. RoverWare integration diagram

4.4 RoverWare Description

Maestro enables users to determine what rover actions they would like to perform next based on the data gathered from the rover through a graphical interface. Users can specify a plan of action through the high-level rover command interface. These commands range from movement commands (DriveToLocation, DriveForward, ChangeHeading), to imaging commands on the available cameras (AcquireImage), and instrument deployment (StowMast, UnstowMast, ManipulatorMove). Different sets of rover interface commands can be specified depending on the functional capabilities of the rover to be

simulated.
The activity plan is then translated into set of high-level commands that are sent to CLARAty, where the actual low-level rover commands are issued to the (simulated) rover hardware. CLARAty's main purpose at this point is to determine how to control the rover based on the rover interface commands. Depending on the technology integrated into the simulated rover, CLARAty has the option to utilize some of its technologies in its mapping from high-level to low-level commands. For example, a simple DriveToLocation command could just be a straight-line drive to the given coordinate. However, CLARAty can also use a variety of obstacle avoidance algorithms to navigate the rover through an obstacle field to the coordinate. From the high-level planning perspective, when the user wants the rover to move to the target, it is the job of CLARAty to get the rover there based using its onboard navigation capabilities.

CLARAty feeds the set of low-level hardware commands it needs in order to achieve its goals to the ROAMS simulation. Based on physical models of the rover and the terrain, ROAMS simulates the dynamics of the rover as if it was executing on real terrain, taking into account physical factors such as coefficients of friction on differing terrains and how rock obstacles affect the wheel chassis. Using a 3D graphics interface called DSpace, ROAMS will display the simulated rover behavior in the given environment to the user (see Figure 5).

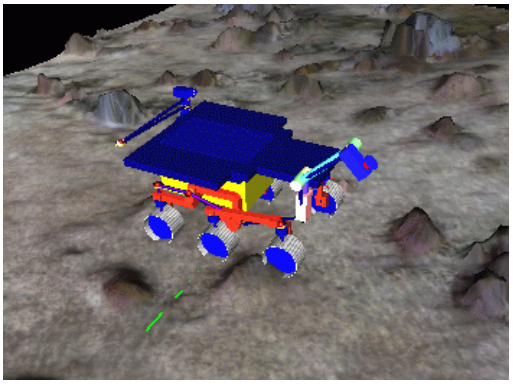

Figure 5. ROAMS "DSpace" view of a simulated rover.

In order to close the loop, ROAMS reports back to CLARAty the success or failure status of individual commands. Based on the algorithm used, CLARAty may choose to re-issue a different command on failure. Also, some high-level rover interface commands may require a product to be returned back to Maestro, such as an AcquireImage command to the Hazcams. It is also the responsibility of CLARAty to package the data product up and send it to Maestro so that it can be processed and analyzed by the scientist users.

5 Ensemble and Agile Development

Ensemble is an open architecture for the development, integration, and deployment of mission operations software. Fundamentally, it is an adaptation of the Eclipse Rich Client Platform (RCP), a widespread, stable, and supported framework for component-based application development. By capitalizing on the maturity and availability of the Eclipse RCP, Ensemble offers a low-risk, politically neutral path towards a tighter integration of operations tools.

5.1 Improving Mission Tool Development

The current approach used to develop mission operations software has produced a set of powerful tools that have enabled stunning successes for NASA. Many parts of the current development process are functioning well and should be preserved. However, improvements in the state of the art in software engineering as well as increasing demands from new missions have exposed several areas that deserve attention. The problems that Ensemble has been designed to address are outlined below.

5.1.1 Brittle Interfaces

Historically, mission operations software has consisted of a set of largely independent tools that communicate with each other using files or socket-based interfaces. The interfaces between the planning tools on MER were a late addition that became the source of numerous problems. Several lessons-learned workshops from MER have identified these interfaces as an area that requires immediate improvement.

File and socket-based interfaces are notoriously difficult to test and debug. As a result, these kinds of interfaces tend to fail often. The most reliable interface between two tools is usually accomplished via direct use of the respective tools' application programming interface (API). The Ensemble approach ensures that many problems in the interface are discovered at compile time.

5.1.2 Too Many GUIs

The number of separate tools used in the MER Activity Planning and Sequencing Subsystem (APSS) is also the source of a popular complaint because it requires mission operators to interact with many different user interfaces in order to get their work done. This slows the overall pace of mission operations and increases training requirements.

The complexity of mission operations makes it infeasible to develop a single operations tool capable of accomplishing all necessary tasks. However, Ensemble's

reliance on Eclipse provides a common GUI framework that can contain GUI components from multiple tools developed by different teams. A mission can then easily reuse any component at multiple stages of the operations process. For instance, a data view that was historically available only during the sequencing phase of operations can be displayed and used at any time if that view is developed as an Eclipse plug-in. The result is a GUI that feels like a single tool to the user, but draws upon the resources of many development teams.

5.1.3 Duplication of Effort

The tools in existing mission operations systems are designed to address the needs of a certain phase of the operations process. One tool is designed to accomplish science planning while another is used for command sequencing. However, certain capabilities are needed at multiple stages in the operations process. Unfortunately, the architectures used in current mission operations tools do not allow capabilities from different tools to be reused at multiple steps in the process. As a result, redundant versions of these capabilities are developed by multiple teams and inserted into separate tools.

This reuse is possible because of the manner in which Ensemble plug-ins deal with the spacecraft plan. In the past, the spacecraft plan has been handed from one tool to the next in a serial fashion. At each step, a single tool had exclusive control over the plan. In contrast, Ensemble plug-ins interact as a group with a common model of an evolving spacecraft plan. Each plug-in can contribute to the plan whenever it is necessary, and each plug-in must respond appropriately to modifications made by other plug-ins.

5.1.4 Lack of Agility

Most development teams strive to make their tool applicable to multiple missions. This is a positive goal because it enables future missions to capitalize on the investment made by prior missions. However, it can also force a mission to accept and maintain capabilities that it doesn't need. The popular "core/adaptation" model is an attempt to insulate different customers from customer-specific requirements, but what if one customer only needs a fraction of the core? Currently, that customer is simply forced to accept the rest of the core anyway, along with the risk and costs associated with its maintenance.

As a multi-mission architecture, Ensemble also supports extensive reuse of components between missions. The vast majority of Ensemble plug-ins are mission-independent, and mission-specific plug-ins are clearly identified. Ensemble is already being used to support Phoenix, Mars Science Laboratory (MSL), and several technology programs, and these customers share a large amount of code in common.

5.2 Agile Development

SAP for MER was built by a small team of developers writing code mostly in isolation. Group meetings were convened only when pertinent issues arose regarding the overall system architecture or an API needed to be designed between components. This approach led to a system where each developer was an expert regarding one specific area of the system but relatively naive regarding other system components. Furthermore, it hindered the ability to perform integration tests on a frequent basis.

In developing the next generation mission operations software, the Maestro team shares a lab with three workstations configured to optimize the experience of pair programming (i.e. two monitors, two keyboards, and two mice per workstation). The lab environment along with the co-development of production code has greatly increased inter-team communication as well as transferred knowledge of the entire code base across all team members. Furthermore, through an adaptation of the twelve tenants of extreme programming, the Maestro team is able to remain responsive to all its customer needs – ranging from class-A missions to individual researchers.

Acknowledgements

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

References

[1] Norris, Jeffrey S., Powell, Mark W., Vona, Marsette A., Backes, Paul G., Wick, Justin V., "Mars Exploration Rover Operations with the Science Activity Planner". Proc. IEEE Conf. on Robotics and Automation, Apr. 2005.

[2] Justin V. Wick, John L. Callas, Jeffrey S. Norris, Mark W. Powell, Marsette A. Vona, III, "Distributed Operations for the Mars Exploration Rover Mission with the Science Activity Planner." Proc. 2005 IEEE Aerospace Conf., Mar. 2005

[3] R. Volpe, I.A.D. Nesnas, T. Estlin, D. Mutz, R. Petras, H. Das, "The CLARAty Architecture for Robotic Autonomy." Proc. 2001 IEEE Aerospace Conf., Mar. 2001.

[4] A. Jain, J. Guineau, C. Lim, W. Lincoln, M. Pomerantz, G. Sohl, R. Steele, "Roams: Planetary Surface Rover Simulation Environment," Intl. Symp. on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), May 2003.