

Component-Based Tool for Mission Operations Software Deployment

A.E. Lindsey*

QSS Group Inc., Moffet Field, CA, 94035-1000

A concerted effort to leverage the experience gained during the Mars Exploration Rover (MER) mission has been initiated by the National Aeronautics and Space Administration (NASA). The intended goal is to redesign, implement, deploy and improve upon several mission-critical spacecraft operations tools. The next generation software that enhances the current suite of MER tools is being developed jointly at NASA Ames Research Center and the Jet Propulsion Laboratory. The resulting collaborative effort has produced an open architecture component-based software tool called *Ensemble* that is used for the development, integration and deployment of mission operations software. This publication presents the *Ensemble* framework, specific challenges of integrating multiple applications into a single platform are discussed as well as the advantages and novelty of our new tool. Finally, a survey describing the process of bundling several MER mission operation software tools into the *Ensemble* application is examined.

I. Introduction

The Mars Exploration Rover (MER) mission is the most complex planetary surface operations mission NASA has ever attempted. In an effort to maximize the amount of daily science activity performed by the rovers, stringent time constraints are placed on MER scientists to analyze incoming data from the deep space satellite network, select science goals for the next day, build an activity plan for achieving the goals and finally put together a sequence of commands that enable the rovers to accomplish these goals. Although the mission has been a huge success, the original planning and operation tools are cumbersome to use and lack process integration. The MER tools are in essence large applications with separate interfaces that are loosely coupled together making analysis, planning and scheduling overly complex and inefficient.

A collaborative effort between Ames Research Center (ARC) and the Jet Propulsion Laboratory (JPL) is developing next generation software that incorporates key component functionality from the current suite of MER tools into a single application. This cooperative endeavor has yielded an innovative component-based software tool that enables different applications to integrate into a common platform and interoperate as a single unit. The resulting open architecture tool, called *Ensemble*, is used for the development, integration and deployment of mission operations software. *Ensemble* is based on the Eclipse Rich Client Platform (RCP), a popular, stable, and well-supported set of Java classes that define architectures for general component-based applications. Any subsequent applications are built on top of the RCP as a set of components called *plug-ins* that augment and extend the platform's functionality. Team members have overwhelmingly elected to use the open source Eclipse IDE, consisting of the RCP together with a set of plug-ins that add capabilities such as compiling and debugging programs, for Java software development. By capitalizing on the maturity and availability of the Eclipse RCP, *Ensemble* offers a low-risk, platform independent means towards a tighter integration of operation tools. Moreover, the chosen approach enables component reuse throughout the operations process, simplifies tool integration by providing a standard development environment and provides a unified GUI for all operation tools.

Currently, our JPL collaborators are redelivering the MER activity planning tools as *Ensemble* components. First round user testing with MER mission personnel has already been performed. The basic integration between planning and sequencing components is complete. A preliminary interface between *Ensemble* and a planner capable of resolving constraint violations has also been demonstrated. The 2007 Phoenix Lander science interface tool is based on *Ensemble*, using components from ARC and JPL. In addition, *Ensemble* has been base-lined for the 2009 Mars

* Computer Scientist, Computational Sciences Division, NASA Ames Research Center M/S 269-2, Moffett Field, CA 94035-1000.

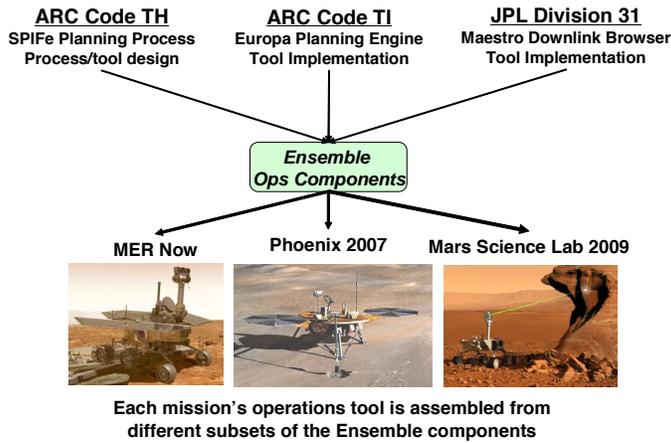


Figure 1. Ensemble Flight Missions.

Science Laboratory (MSL) activity planning function. The software for all three missions will be maintained and developed on a single platform (see Fig 1.).

Based on the MER mission experience we believe future space operations will greatly benefit from four basic tenets adopted by the Ensemble platform methodology:

- 1) Divide large applications into interoperable and reusable components.
- 2) Deliver to a mission only

those components it needs.

- 3) Allow reconfiguration so the work process can evolve as experience is gained.
- 4) Usage of continuous automated build and test systems.

The paper contents are organized as follows: An overview of the Ensemble plug-in architecture, MER activity planning and sequencing subsystem, MER mission problems addressed by Ensemble, a discussion of how Ensemble components are being used in space missions, future directions and conclusion.

II. Ensemble Plug-in Architecture

Ensemble uses software components called plug-ins for command sequencing, science planning, plan representation, activity planning and modeling see Fig. 2. The underlying architecture of Ensemble is Eclipse which is both an application framework and a Java-based IDE. Ensemble participants working for a particular customer project create and test software components in the form of Eclipse plug-ins that are integrated into an Ensemble product (or products) for that customer. An Ensemble product is built from the Eclipse Rich Client Platform (RCP) and a selection of shared Ensemble plug-ins, plus a selection of customer-specific plug-ins.

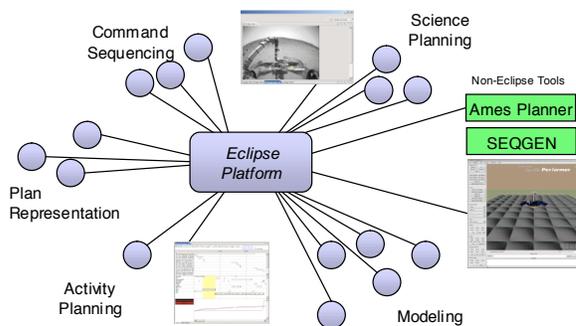


Figure 2. Ensemble Plug-in Architecture.

Plug-ins within Ensemble fall into two categories: "owned" and "shared":

- Owned – Each Ensemble participant will develop some capabilities that fall within an area that they consider their principal focus. The plug-ins that make-up these capabilities are considered to be exclusively owned by the participant and are clearly marked in the Ensemble repository with the reverse-domain name of the team at the beginning, e.g. gov.nasa.arc.teamname.plugin. The plug-in source code is always available for reference by other Ensemble participants, however all modifications are submitted to the participant that owns the plug-in. These modifications are also considered the property of the participant. In other words, contributions to an owned plug-in do not grant the

contributor any ownership of the plug-in. Ensemble participants may use an Owned plug-in in a project only with the permission of the participant that owns it.

- Shared – Some capabilities (for example, parsing an activity dictionary) are utilitarian in nature and should be developed only once and maintained centrally. The plug-ins that comprise these capabilities are developed cooperatively by Ensemble participants and are marked in the Ensemble repository with names of the form gov.nasa.ensemble.plugin. All Ensemble participants may modify and use the plug-ins for any purpose, regardless of whether they contributed any code to its development.

The Eclipse RCP performs the following functions:

- Hosts an application's plug-ins, managing the interface between plug-ins.
- Provides access to key parts of the Eclipse User Interface, including menus, toolbars, perspectives and status displays.
- Provides access to plug-ins with other useful capabilities, including an integrated help system, an automatic update manager, graphical editors, charting and graphing.

The applications that are currently planned for development within Ensemble deal with science planning, and activity planning and scheduling. A typical application program or set of application programs built from Ensemble plug-ins might perform one or more of the following functions, each with direct user involvement:

- Search for and display images needed for planning.
- Enable the user to define a target (for an imaging activity or a traverse activity) by clicking on an image.
- Define an instance of an activity by selecting from an Activity Dictionary and specifying values of parameters for the instance.
- Specify temporal relationships among activity instances.
- Display a timeline of activities and of groups of activities; enable a user to change the scheduled time of activities by dragging items on the timeline.
- Call upon external models to simulate the plan and display results textually and graphically.
- Call upon automated planners to schedule activities into legal positions.

Many of the possible applications planned for Ensemble overlap, in the sense that they share plug-ins. For example, plug-ins that display lists of activities to the user are present in any application that needs to display lists.

Some plug-ins, referred to as core plug-ins, can be distinguished, although architecturally from an Eclipse point of view, there is no distinction. The core plug-ins are expected to be a constituent of several Ensemble-based applications because they offer functionality that is common in planning applications. Sometimes it is clear at the beginning of the development of a plug-in that it should be considered core. Other times, a specialized version for a single project's definite need might first be implemented, and then later its general features factored out to become a core plug-in. Functions that are specific to a single project are implemented as non-core plug-ins. Thus, a specific application consists of the RCP, some core and customer-specific plug-ins.

III. Activity Planning and Sequencing Subsystem

The surface operations' planning for the MER rovers has been ambitious and challenging. The daily rover activity planning process must balance different science and engineering needs, while taking into account resource limitations, operations safety rules, etc. The Activity Planning and Sequence Subsystem (APSS) is a part of the Ground Data System and a critical component for the mission operations rover uplink process. It is responsible for strategic mission planning, science observation planning, and engineering activity planning and sequence command uplink. APSS supports both user and automated reasoning operations needed to modify activity plans. The system uses a vast array of software tools during various mission phases. Some of these applications include returned data analysis, resource estimation; spacecraft command preparation and activity sequence planning as seen in Fig. 3.

The following is a list of APSS components:

- Europa – an artificial intelligence based planner, developed by ARC.

- APGEN – the activity plan generator resource model-based planner developed by JPL.
- SAP – the science activity planner is used to facilitate science telemetry analysis and activity planning.
- RSVP – the rover sequencing and visualization program is used for advanced Martian surface and rover visualizations as well as to generate the actual rover command sequences.
- SEQGEN – the sequence generator predicts events that will occur on the spacecraft as result of the sequence, giving warnings when the sequence violates *rules* or causes spacecraft subsystems to be misused.
- SLINC – the spacecraft language interpreter and collector is responsible for the translation of a spacecraft sequence in the form of a spacecraft sequence file (SSF) into a command packet file (CPF) for broadcast to the spacecraft. In addition a binary UNIX file may be formatted into a CPF for transmission to the spacecraft.
- RSFOS – the re-engineered space flight operations schedule is a program that reads user maintainable ASCII tables and an input predicted events file (PEF) generated by SEQGEN. One of the output files is input to the graphical editor, “soeedt”, used to generate an operations schedule.
- CAST – the common allocation scheduling tool is an integrated set of multi-mission software tools that assist projects in negotiating deep space network (DSN) coverage.
- SEQREVIEW – the sequence review tool re-formats and extracts information from sequence products (e.g. PEF) which allows users or external applications to analyze the data.

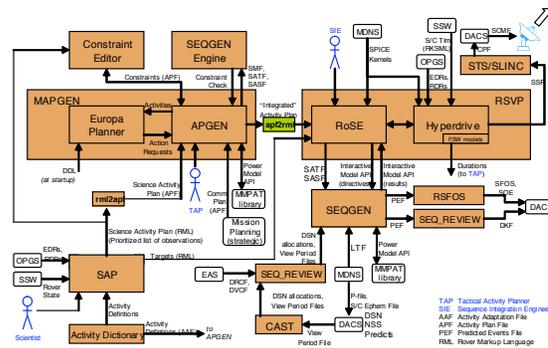


Figure 3. MER Nominal APSS.

The two primary goals for the GDS are

- 1) Efficient and flexible activity planning for large-scale, complex tasks.
- 2) Usage of an integrated system for analysis, planning & execution.

The originally designed APSS proved ineffective at efficiently meeting these goals. In particular, the subsystem consisted of loosely connected and difficult to monitor scripts functioning to mirror large portions of the GDS. In addition, the interfaces were brittle and linked together with ordinary flat files and the infrastructure lacked overall process integration. The usage of this original system presented many mission planning challenges.

IV. 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 software engineering as well as increasing demands from new missions have exposed several areas that deserve attention. A redesigned APSS using Ensemble components has been developed (see Fig. 4) and successfully delivered to MER. The new APSS has been designed to address a number of problems areas outlined below.

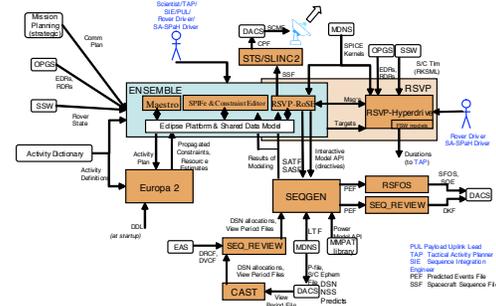


Figure 4. MER Ensemble APSS.

A. 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. Consequently, this area 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). This approach ensures that many problems in the interface are discovered at compile time.

Direct API interfaces can be very difficult to implement when the tools being integrated were developed in different environments. Since most Ensemble members agree to develop their tools as Eclipse plug-ins, these issues are minimized. Moreover, Ensemble draws upon capabilities provided by the Eclipse RCP to document and enforce interfaces between different components.

Components can still be integrated with the Ensemble architecture despite not being developed as an Eclipse Java plug-in. Work is underway to develop a general, robust method for non-Eclipse tools to interact with other Ensemble tools. By reusing a single interface point for multiple external tools, it is expected that the reliability of these interfaces will be increased.

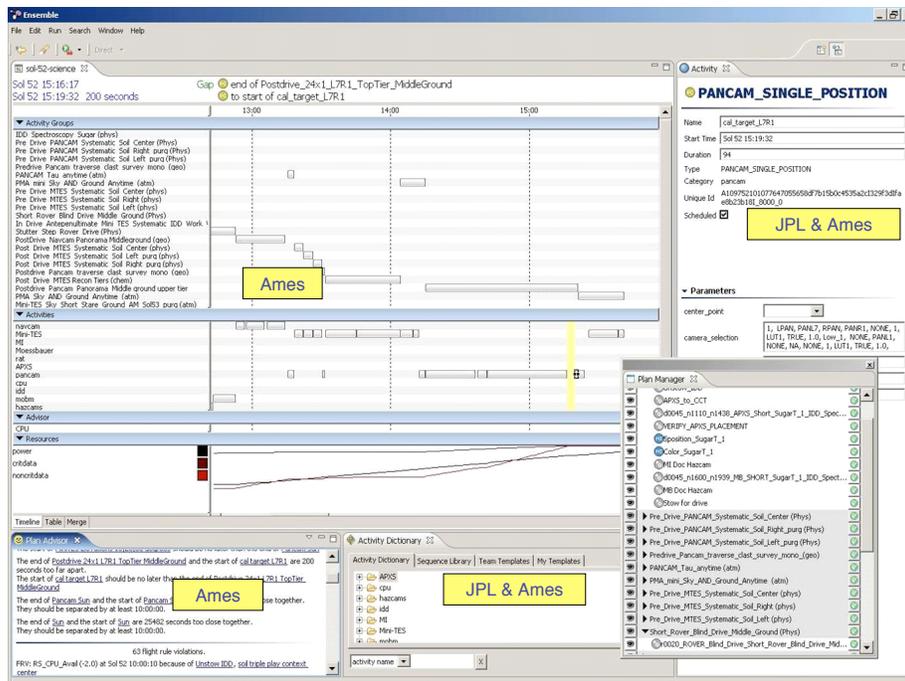


Figure 5. Planning Perspective.

B. Excessive number of GUIs

The number of separate tools used in the MER APSS is also the source of notorious complaints 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.

Ensemble uses a task-oriented GUI that is based heavily upon Eclipse *perspectives*. A perspective defines which GUI components will be visible to a user at a particular time. As a user moves through the tasks required for planning, they click through a set of icons devoted to each task. For instance, a user might click on an icon to

activate a perspective devoted to browsing downlink data (see Fig. 5) and then move onto a perspective devoted to building a set of activities for planning (see Fig. 6). Ensemble’s planning perspective allows activities to be arranged into a plan, browsed and edited, and linked together with constraints. Users can browse downlink data, manipulate it, and create targets using the data browsing perspective. In fact, we can mix and match components from the perspectives, to show the details of an activity next to an image illustrating the target of the activity for example.

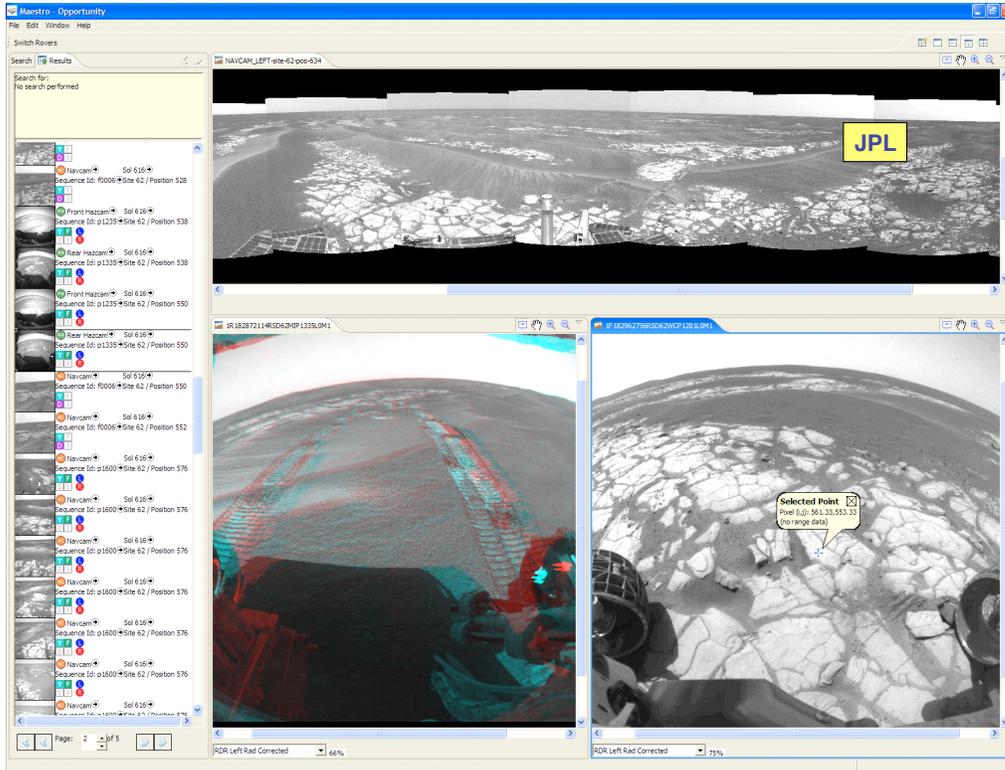


Figure 6. Downlink Perspective.

C. Duplication of Effort

The tools in existing mission operations systems are designed to address the needs of a certain phase of the operations process. 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.

The component-based development model and the perspective-based GUI that Ensemble inherits from Eclipse enables a mission to 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.

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.

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, MSL, and several technology programs, and these customers share a large amount of code in common.

D. 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.

An Eclipse (and therefore Ensemble) application consists merely of the core RCP plus a set of plug-ins. The set of plug-ins that are included completely define the functionality supported by the program. Because plug-in dependencies are clearly documented and checked through the Eclipse IDE, it is possible to release only those plug-ins to a customer that provide the functionality they need. For instance, if a particular customer doesn't need spectral visualization support, the plug-in(s) that provides that capability can simply be omitted from their release. This provides a tool that is easier for the customer to test and learn to use.

V. Ensemble Mission Components

Ensemble components are actively being used as part of the MER APSS. Ensemble has integrated the following five powerful engines (Maestro, SPIFe, Europa, RoSE and SEQGEN) into a single application. The Maestro Ensemble component subset is the next generation SAP and has been delivered to the MER science team for beta testing. The MER science and uplink team members have estimated that overall science return increased by 20 – 50% during the beta testing phase. Additional components are actively being integrated into Ensemble, e.g. high fidelity terrain modeling and analysis.

Ensemble is also being feature developed for two future space missions. In particular, Ensemble components have been designed for usage in the Phoenix 2007 (see Fig. 7) and MSL 2009 missions (see Fig. 8). Phoenix is a long-armed Lander designed to examine the icy plains of northern Mars for potential habitats in water ice, and to search for evidence of life, past or present. The Phoenix activity planning tool called the Phoenix Science Interface (PSI) includes the Ames Ensemble component subsets SPIFe and Europa as well as the JPL Ensemble component subset Maestro. Currently the term SPIFe is used loosely to refer to a collection of interfaces, principally a timeline, that are used during activity planning. In fact, both the SPIFe planning interface and the Europa planning engine are two reusable mission operations tools.

The MSL mission will deliver a mobile laboratory to the surface of Mars to explore a local region as a potential habitat for past or present life. The rover will be equipped with an impressive array of scientific instrumentation and its own power source. It is highly probable that MSL will use the current MER mission GDS. Consequently, the delivered Ensemble components and re-designed APSS that already augment the GDS with greater efficiency and

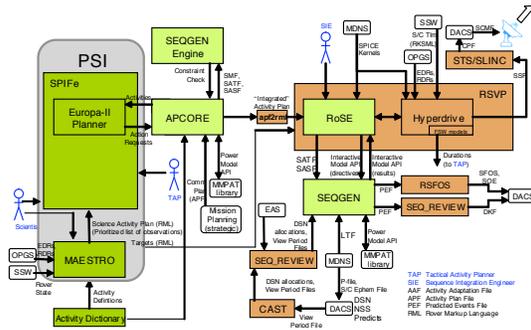


Figure 7. Phoenix Ensemble Components.

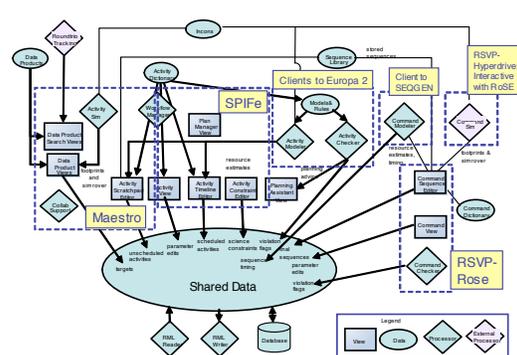


Figure 8. MSL Ensemble Components.

flexibility is expected to demonstrate similar success in the future missions.

VI. Future Directions

When science and engineering teams navigate the two MER rovers across the roughed Martian terrain, several hundred people collaborate to develop daily activity science plans. Teams typically collaborate in different modes including face-to-face interactions and meetings requiring parties to participate at the same time but distributed geographically. Needless to say the efficiency with which teams access and share information directly impacts the amount of science data returned.

NASA ARC has developed a large touch-screen based collaborative computer called *MCCBoard* that has been used to assist science teams during MER surface operations. MCCBoard's large interactive work surface facilitates collaboration among planning teams that can gather around a five foot long plasma display board to retrieve, view, share and annotate mission data and rover images. MCCBoard is designed for extendibility and uses an Eclipse plug-in architecture. The architecture supports functionality enhancement and customization of its core facilities.

The *Mission Control Technologies* (MCT) project is a new framework for the creation of mission systems using collaborative component technologies. Currently MCCBoard is being redesigned and implemented using a collaborative component technology infrastructure. The resulting next generation software intends to offer enhanced collaborative capabilities such as virtual file storage, Open Document format support, chat, instant messaging, reconfigurable board tool views, etc. It is believed that Ensemble in concert with such collaborative software would greatly aid surface operations planning for future space missions.

VII. Conclusion

Automated and mixed-initiative, systems that integrate human and automated reasoning, decision support tools have been critical to the success of MER. One serious limitation of the MER operations tools has been that they were developed separately rather than in an integrated fashion. The Ensemble architecture enables NASA missions to derive greater results from their investment in mission operations software. Rather than arranging together a series of largely isolated and independent tools, missions are able to seamlessly assemble precisely the tool they need by drawing components together from different development teams. As a result mission operators are more efficient, ultimately improving the overall performance of our missions. Finally, operations software developers are better able to focus more on developing great tools and less on frustrating integration issues.

Ensemble is built on the Eclipse Rich Client Platform and a selection of plug-ins. The open source platform benefits from broad participation among multiple teams from different NASA centers. Each team is assigned a specific Ensemble component subset to develop. The Maestro component subset has been delivered to MER for beta testing. Moreover, two future space missions have been base-lined to use Ensemble as an integral part of their activity planning and sequence subsystem.

A successful collaborative distributed mission requires more than just tools that play well together. It also needs frameworks and interfaces that facilitate such interoperability. The MCT project is developing software technologies and design methods for modular, evolvable, distributed ground data systems and mission operation systems to meet those needs. The MCCBoard operates at several NASA centers and supports the MER mission. Currently the board software is being redesigned and implemented using MCT components. A number of space missions would undoubtedly benefit from using Ensemble and collaborative component technologies. Particularly those missions involving surface operations activity planning and large distributed science teams.

Acknowledgments

The research in this paper was carried out by NASA Ames Research center and the Jet Propulsion Laboratory, California Institute of Technology, under a contract with NASA. The author would like to thank James Kurien, Jay Trimble and Jeff Norris for their material support.

References

Proceedings

Norris, J. S., Powell, M.W., et. al. "Science Operations Interfaces for Mars Surface Exploration," *IEEE Conference on Systems, Man and Cybernetics*, IEEE, Big Island, HI, 2005.

Ko, A., Mالدague, P., et. al. "Design and Architecture of Planning and Sequence System for Mars Exploration Rover Operations", *SpaceOps Conference*, 356-434, Montreal, Canada, 2004.