

Distributed Operations for the Mars Exploration Rover Mission with the Science Activity Planner

Justin V. Wick, John L. Callas, Jeffrey S. Norris, Mark W. Powell, Marsette A. Vona, III
Jet Propulsion Laboratory, Pasadena California
Justin.Wick@cornell.edu

Abstract— The unprecedented endurance of both the Spirit and Opportunity rovers during the Mars Exploration Rover Mission (MER) brought with it many unexpected challenges. Scientists, many of whom had planned on staying at the Jet Propulsion Laboratory (JPL) in Pasadena, CA for 90 days, were eager to return to their families and home institutions. This created a need for the rapid conversion of a mission-planning tool, the Science Activity Planner (SAP), from a centralized application usable only within JPL, to a distributed system capable of allowing scientists to continue collaborating from locations around the world.

Rather than changing SAP itself, the rapid conversion was facilitated by a collection of software utilities that emulated the internal JPL software environment and provided efficient, automated information propagation. During this process many lessons were learned about scientific collaboration in a concurrent environment, use of existing server-client software in rapid systems development, and the effect of system latency on end-user usage patterns.

Switching to a distributed mode of operations also saved a considerable amount of money, and increased the number of specialists able to actively contribute to mission research. Long-term planetary exploration missions of the future will build upon the distributed operations model used by MER.

TABLE OF CONTENTS

1. INTRODUCTION	1
2. CENTRALIZED OPERATIONS.....	1
3. MOVING TO DISTRIBUTED OPERATIONS.....	2
4. ARCHITECTURE AND IMPLEMENTATION.....	5
5. TECHNICAL CHALLENGES	6
6. MISSION IMPACT	6
7. RECOMMENDATIONS FOR THE FUTURE.....	7
8. CONCLUSION.....	8
REFERENCES.....	8
BIOGRAPHY.....	8

1. INTRODUCTION

The Mars Exploration Rover Mission has been an unqualified success. At the time of this writing, both the Spirit and Opportunity rovers have exceeded their operational lifetimes by a factor of three. Both rovers continue to roam Mars, returning a wealth of valuable new information to Earth.\

The unprecedented length of the Mars Exploration Rovers mission created many challenges for mission planners.

Although the original architecture of the mission planning system was intended to be distributed in nature[1], budget constraints did not allow for the development of this capability. As a result, the planning software was designed such that it was heavily reliant upon internal computing resources at JPL, making it unusable at remote sites.

In March 2004, as the primary mission for both rovers drew to a close, it became evident that both rovers were likely to continue operating long past their original 90 sol lifetimes. Faced with the reality that mission scientists would shortly begin departing from JPL to their respective research institutions, the decision was made to redesign the system to accommodate the participation of scientists at remote sites. Since the Science Activity Planner was the primary tool used by scientists in the high level planning process, an effort was begun to adapt SAP for use outside of JPL and to provide a collaborative framework for scientists to operate it.

2. CENTRALIZED OPERATIONS

During the centralized phase of the MER mission, scientists were gathered in a single building at JPL and used the planning and analysis software in a specially configured computing environment, which was called the Flight Operations System. The Flight Operations System was the platform for the Ground Data System (GDS) software that drove the data processing and planning processes for the mission.

The Science Activity Planner (SAP) is a GDS tool that is used to perform dual roles facilitating manual science and engineering-level data analysis, and planning the daily actions of each rover in a coarse, high level fashion. This tactical decision-making process is similar to that practiced in the FIDO field tests [2]. SAP provides functionality allowing users to view and manipulate images captured by the rover in what is known as the Downlink Browser (figure 2). There, objects can be examined, and points in space known as “targets” can be marked for use in creating planned activities. The Uplink Browser (figure 3) allows users to create “activities” (a high level description of an action the rover should take, such as acquiring an image or driving) and “observations” (groups of related activities).

Each day the tactical process starts with a science meeting during which scientists are briefed about the current situation. After this, the scientists break up into “theme groups” such as “atmospheric” or “soils” or “long term planning” and work in parallel, using SAP to construct

sequences of instructions for the rover that reflect their scientific goals. During this several hour period, the data is analyzed within SAP, points are designated as targets for the rover's actions, and the potential plans are put together.

After this time, the final plan is debated and assembled in the Science Operations Working Group (SOWG) meeting. Possible scientific observations from each group are ranked according to importance. After a structured debate, the accepted observations are arranged together in SAP to meet the daily energy, time, and bandwidth budgets that have been established by the engineering team. The final merged plan is then delivered for further refinement and processing downstream to be converted to the actual sequence of instructions sent to the rover.

A homogeneous computing environment consisting of custom-built workstations running the Linux operating system facilitated collaboration within the system. There was a central Network File System server (the OSS) for each rover (MER-A and MER-B) and also a central SQL database server for each. Because all downlinked data and planning information were kept in these two central repositories, collaboration was simple. When a scientist saved a plan file, it was immediately available to all others to be analyzed and merged. Target designation, critical to the planning process, was also synchronized with low latency (~2 sec) via the central SQL server. All science workstations were guaranteed to have access to the exact same set of data.

3. MOVING TO DISTRIBUTED OPERATIONS

Due to budget and lifestyle constraints, the mission was shifted to a new, more distributed mission architecture. The cost of keeping relevant scientists on location in Pasadena was prohibitive, and many of the scientists and engineers had family elsewhere in the world for whom they were responsible. Because of this, the decision was made to create an environment in which scientists could tactically plan with SAP at remote sites. This software environment would have to:

- Make planning-relevant downlink data available to the remote scientists in a timely fashion.
- Allow scientists to interactively share targets designations.
- Facilitate the sharing of plan files that contain the scientific observations for the day.
- Dynamically create indexing metadata of available data products to make them available in SAP.
- Maintain operational security through use of encryption, authentication, and firewalls.

It was decided that the best possible action was to closely replicate the JPL software environment, rather than change SAP itself. SAP expects a highly structured file system database containing images, range data, three dimensional meshes, spectral data records, coordinate frame information, planning constraints, and plan files. Because no available network file system server was fast enough to be used by SAP interactively, the relevant data sets would have to be mirrored locally. This also meant that the indexing of that

data (which is how SAP knows what information is available on the file system) would also have to be done locally. Also, the sharing of targets and plans presented a challenge, since the servers hosting the plans and targets were not accessible outside of JPL.

Data Synchronization

The first matter was to arrange for the data to be delivered to the remote SAP workstations. SAP expects data to exist in a highly structured, hierarchical system of folders, numbering well over a million for each rover. This file system database is known as the Operational Software System, or OSS. The folders separate data by sol (martian day), instrument, and data type. The job of the data synchronization subsystem was to replicate the internal file system database of downlinked data on client workstations around the world.

The first solution to this problem that was developed utilized an open source program known as RSYNC, which can synchronize files and directories recursively between machines through a secure SSH tunnel. A daemon was created that repeatedly synchronized the directories for recent sols with a central server. The central server itself was to be filled with the SAP-relevant data from the operational NFS servers.

The problem with this approach was that it relies heavily on polling, and tens of thousands of files and directories had to be recursively compared. It was decided that it would place too much of a load on the server to have an acceptably low latency for data delivery. Worse, overloading issues were already a severe problem on the operational NFS server, and it was decided that this solution would most likely exacerbate the situation.

We then decided to create a second data synchronization solution, utilizing the JPL Multi-mission Image Processing Lab's (MIPL) File Exchange Interface (FEI). FEI is a system that MIPL uses to automatically push out data to remote sites, such as research institutions or museums. While it supports polling and client-initiated downloading, it also has an event-driven server-push mode that relies on the "subscriptions" of a client to a set of file types. FEI was chosen because it has a very low latency (~2 seconds within the JPL network) and is very well load balanced.

The main problem associated with this approach was that FEI does not keep track of the path in the file system to the directory where a particular file came from – this data would have to be reconstructed. In addition to this, FEI also contains a large number of files that cannot be used by SAP and are not relevant for tactical planning. Because of the low bandwidth at many remote sites, the files would have to be filtered for relevance prior to downloading. The system also needed to allow for the gathering of archived files from specific sols of interest – a feature not supported by FEI.

The final solution was to have two methods of getting files – an automatic subscription program, and a manual archived

file retrieval program. Both programs used a filter to determine whether or not a given file was desired based on its relevance (and in the case of archival data, whether or not it fell within a specified range of sols). Also, a script was assembled that could sort the files into their final locations based solely on the file names. This was made possible by the fact that the file names systematically encode the data type, instrument name, time acquired, and the rover from which the data was obtained.

Each workstation established a connection with the FEI server, and signed up to be “notified” when files in a relevant “filetype” were made available. This notification was pushed from the server to the client, at which time the client decided, based on the filename, if the file was desirable. If the file was wanted, it was retrieved from the server and then sorted into the file system. This system has latencies on the order of minutes or less, and has nearly ideal bandwidth use (the server/client messages are very short).

Obtaining access to archival data was somewhat less straightforward. That program, given a rover designation (A or B, for Spirit or Opportunity) and a desired range of sols, downloads an entire roster of all available files in relevant filetypes. It then filters the names of files to find those files that fall into the specified range of sols, and also do not currently exist on the local file system. This roster listing process is very inefficient and takes several minutes, however downloading the requisite data can take hours, so the overhead is acceptable.

The final step in the data synchronization process is the Data State Manager Daemon – a daemon process that scans available downlinked data products and creates a comprehensive index of what data is available. Every thirty seconds the most recent sols are scanned (and occasionally older sols, according to a probabilistic algorithm) to see if new data has been made available. When new data is discovered, it is processed and incorporated into the index, making it available for SAP. A nearly identical process is run at JPL, where the cost of all open SAP instances scanning each sol would have been prohibitive.

Target Synchronization

Target synchronization was another vital component of the distributed SAP system. At JPL, targets were synchronized between machines by storing them in a central SQL server. The various SAP instances would poll the server every two seconds, checking timestamps in the database to see if new targets had been created, or if old ones had been modified. There were no security issues because the database was not accessible from the outside world, and all individuals using computers that could access the database were cleared to designate targets.

In a distributed setup, however, everything changed. It was no longer possible to make the central JPL target server available to machines outside of JPL for security reasons; however, each remote site had to be able to see the same targets as users at JPL with minimal latency. Moreover, there had to be a method to take targets from outside JPL

and import them into the internal JPL server. This entire process was required to be as low-latency and automatic as possible, while maintaining operational security.

The solution we chose was that there should be a secondary, “external” SQL server that would be accessible to authorized machines outside of JPL. A script at JPL forwarded changes and new additions to the JPL internal target database out to the external server every few seconds. Because of the nature of the database, it was acceptable for targets to exist in the external database but not in the internal database without causing any problems. Plan files, however, contain reference to targets (to decide where to drive, or aim a camera, etc). If a plan were brought into JPL that referenced an external target, that target would have to be manually imported by a script at JPL. That script would then have to extract a static copy of the target from the plan file text. This process was considered secure because it required a human in the loop to verify that the target was valid. Also, the external server was protected by a strict firewall that only allowed access from a set of secured university computers that were certified as part of the planning process. The data from JPL was encrypted using an SSH tunnel with public key authentication.

A final consideration for target sharing was the complication that was caused by SAP’s use of MySQL database polling. The newly changed entries in the JPL target database had to have a timestamp in the remote database that would cause the remote SAP clients to notice the change. Due to various internal details of the SAP client and MySQL servers, these timestamps had to be adjusted into the future before being sent to the external targets server. This caused new targets to be downloaded repeatedly by clients, but the bandwidth that this used was acceptably small.

Plan Sharing

The issues associated with plan sharing were similar to those of target synchronization in that the central server (in this case, the internal NFS server at JPL) was not accessible to the outside world. Also, there were similar security concerns. Plan files coming out of JPL automatically were not considered to be a security issue. However no one from outside JPL could be able to insert a plan file into the normal planning directories inside JPL.

The solution chosen was that there should be two repositories for plan files, one inside JPL (the NFS server) and one outside JPL. These two repositories would automatically synchronize, however no user outside JPL could be allowed to write a file that would propagate to a normal planning directory inside JPL. Instead, users outside JPL would have to place plans into special “external” directories. The planning directories inside JPL for each sol had names such as “apxs” or “soil”, etc, broken down by group, and each containing an additional named “working” directory. The planning directories were modified by adding another directory named “external” in each subgroup directory. A user outside JPL could submit his plan to the central external plan server, but only if it resided inside an “external” directory.

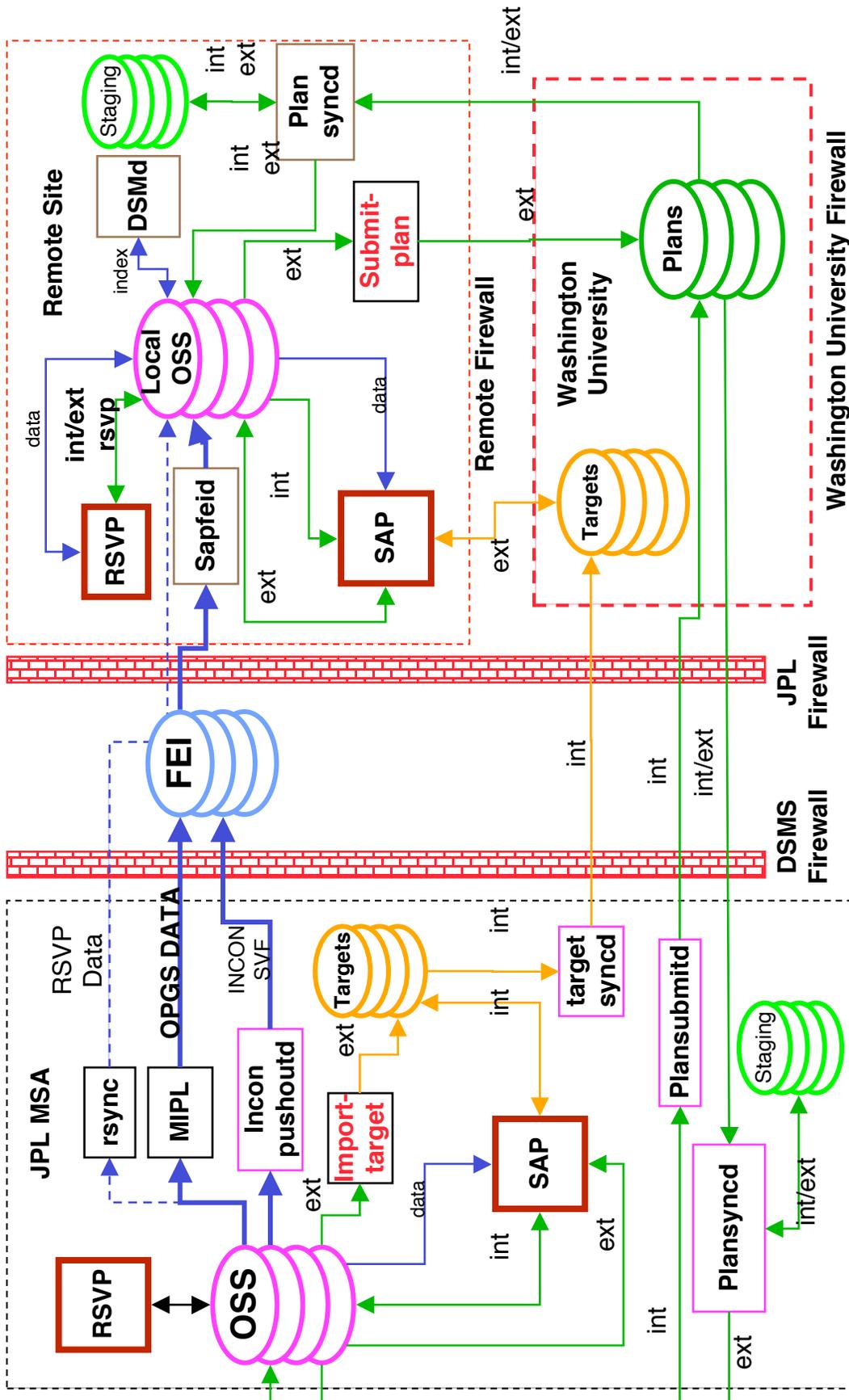


Figure 1 – Data flow in Distributed SAP. Blue arrows indicate data products. Green indicates plans. Orange indicates targets. Processes marked in red require human intervention.

The majority of the synchronizations were automatic. JPL's NFS server was considered the canonical source for "internal" plans. Every 30 seconds the next 5 sols worth of internal plans were sent to the external server. Every 30 seconds or so, those same sols were synchronized from the external server to the SAP workstations at each institution. However, because there was no single canonical source for plans created at an institution, it was decided that submission of an "external" plan to the central server would be a manual process. Once an "external" plan was submitted to the central server, within 30 seconds it would be copied to the same directory on the JPL NFS server, to be seen by those at JPL. This is how planned observations that were created outside of JPL could become part of the final plan at the SOWG meeting held at JPL.

4. ARCHITECTURE AND IMPLEMENTATION

Figure 1 illustrates the overall architecture of the system. The left side of the diagram represents the portion of the system running at JPL. The lower right corner of the diagram shows the servers at Washington University of St. Louis. Finally, the upper right represents each individual Distributed SAP workstation. The data flow is illustrated by colored arrows: blue for downlink data, green for plans, and yellow for targets. The cylindrical shapes represent servers, and the named rectangles signify a process or collection of processes that are logically grouped together. A name in red signifies that the process requires a human intervention. Whether or not a target or plan being transferred was created inside or outside JPL is indicated by an "int" or "ext" label on the associated arrow. Names ending in "d" refer to "daemon" processes that run constantly in the background. RSVP is an engineering level planning program that is used by some scientists remotely, and uses much of the same data as SAP.

Downlinked Data

To understand how the system works, one should first examine the downlink data flow (blue arrows). The Multi-mission Image Processing Laboratory (MIPL) is the source of all processed imagery used in this system. MIPL, along with the "Inconpushoutd" – a daemon that pushes out initial conditions of the rover for a sol, the planning constraints, and coordinate frame information – supplies the FEI server with the files that are needed for the use of SAP outside of JPL.

After the files are sent to the FEI server, the clients are notified of the newly available files through the "Sapfeid", a daemon that handles all of the processes that wait for new files. If the files are deemed relevant, they are downloaded from the server into a temporary directory, and then stored in their proper location in the local OSS (the local set of folders that hold the data for each sol). Once the new data is noticed by the DSMd (Data State Manager daemon), it is then indexed. After that the data can be accessed by SAP.

Target Data

The target data flow is more symmetric – a target can originate either at JPL or at an external workstation. SAP instances at JPL create targets in the internal database. A JPL computer running the "targetsyncd" – the target synchronization daemon – takes newly generated targets and sends them out to the centralized target server at Washington University. Every time an external SAP client opens a plan from a given sol, it fetches the targets associated with that sol from the central server. The SAP client also maintains a polling thread that keeps looking for new targets being made on that sol.

A computer external to JPL can create a target in the external database, making it available to all other external SAP instances. If the target needs to be used inside JPL, an external plan file is saved and then submitted to the plan server; a copy arrives at the JPL OSS. A person, either at JPL or logged in remotely then runs the import-target script, giving it the plan file, and the name of the target to be imported. The import-target program reads the target data from the plan file and then enters it into the local JPL database. Any open internal SAP instances can then see the new target, and it can be used in the final, official plan.

Planning Data

The final component of the system is the shared planning data flow (green arrows). Just like shared targets, there are two separate places where plans can be generated – internal to JPL by SAP, or external to JPL by SAP. Inside JPL, they are kept in special directories on the OSS. An automated process called Plansubmitd polls the OSS every 30 seconds to check for new plans, or newly modified plans, and uploads them to the external planning server at Washington University. A similar process called Plansyncd polls the server for new or newly modified external plans to be imported. Plansyncd imports all changed plans to a staging area, but only copies external plans to the actual OSS for security reasons. This prevents anything submitted to the external server from affecting the internal plans without intervention from a human at JPL.

The right side of this data flow concerns the remote sites. If a plan is created or modified at a remote site, and the user wants to share it with the rest of the distributed SAP users, the "submit-plan" script is run. This sends the file to the server (overwriting any older version of that file if it previously existed). Also, a slightly different version of the Plansyncd is running in the background. It is identical to the JPL version, except that it copies both internal and external plans to the local OSS.

Programming Languages Used

All of the daemon programs were written in Perl 5, and utilized utility shell scripts. The import-target program is a combination of a Perl frontend and a Java backend. Perl was used because the system is tied heavily to the underlying OS, and it made invocation of Unix commands

and file manipulation particularly easy. Also a large amount of the work done by these programs involved text parsing.

5. TECHNICAL CHALLENGES

The technical challenges of this project were many and varied. Most of the challenges involved reliable communications between all of the parts of the system, atomicity of transactions, and server load. Also, out of necessity, many parts of the system used software in ways that were not originally intended.

The most common technical challenge of the entire project was the large set of problems created by repeated polling of file systems and servers. Because the MER GDS has no centralized, common event-driven architecture, most of the components of the distributed system use some form of polling to handle propagated changes. Polling itself is not a significant challenge in software development. However, the efficiency of the polling was a severe limiting factor in what design choices were available, and it forced us to use nondeterministic algorithms for some of the less important parts of the system.

Our data indexing process, the Data State Manager (DSM), needed to poll tens of thousands of subdirectories of the file system every thirty seconds. This grew to the point where it was untenable, so a compromise was made in the system's design. Instead of scanning all sol data directories every 30 seconds, it would scan only the three most recent for new data constantly. The older directories would have a probability of being scanned each 30 second sweep such that about 95% of all sols would be scanned in a given 24 hour period. The use of nondeterministic algorithms was considered safe because older sols tended not to change often, and their changes tended not to be important.

Another example where polling was a bottleneck was the Plan Synchronization Daemon. The Plan Synchronization Daemon (plansyncd) relied heavily on polling of a central server. Plansyncd used the RSYNC client tunneled through SSH, and RSYNC only permits one directory to be recursively synchronized per connection. Because of this and the fact that the first five upcoming sols had to be synchronized every thirty seconds, each requiring a separate connection, the SSH authentication server on the central planning server became intolerably slow. While plans still propagated, it was at a reduced rate, and often connections to the server were rejected due to the overload. As of this writing, we plan to replace this polling process with a manual process due to the incredible load it places on the server.

A different issue encountered was reliable communications through a highly heterogeneous network environment. There were a lot of very complicated firewalls involved – two levels at JPL, at least two at Washington University of Saint Louis, and usually between one and two firewalls at other institutions. SSH tunneling made communications

through these firewalls possible, however this required authentication keys to be distributed. Network failures were not entirely uncommon, and temporary workarounds had to be set up in the event that a server was not reachable. Server load and reliability was often the deciding factor for the success of the Distributed SAP system.

One of the biggest causes of bugs was the relative heterogeneity of systems running SAP outside of JPL. Inside JPL the software was run exclusively on Red Hat Linux 7.3 boxes, all of which contained identical processors and graphics cards. Outside of JPL, Red Hat Linux 7.3, 8, 9, Red Hat Enterprise Linux 3, and Fedora Core 1 were in use. This was a problem because it required different systems to use different versions of the FEI client, which was not fully tested on Fedora Core 1. Also, newer Linux distributions shipped version 5.8 of Perl, which has subtly different semantics for a few very important operations, such as regular expression matching. This led to a few bugs involving data delivery, which were very difficult to track down.

Last but not least was the fact that some software components of the system were being used in ways that their creators had not intended. This sometimes put the system into odd states requiring manual intervention. The FEI server system was not designed, for instance, to notify clients if a file that already existed on the server was modified, only when new files were added. So, when certain important configuration files had to be pushed out, they had to be removed and then added to FEI. Also, FEI had no method of filtering files based on the sol they belong to, or specific details of the file type. This had to be implemented in one of the more complicated Perl programs that we created. The same goes for the lack of file system metadata preservation in FEI. The files had to be sorted by a Perl program, based solely on the name of the file – a fact that precluded the sorting of certain types of files accurately.

6. MISSION IMPACT

The impact of the Distributed Science Activity Planner on the MER mission was very significant. By allowing scientists to analyze data and collaboratively plan at remote institutions, Distributed SAP was a primary enabling factor in the feasibility of the distributed operations architecture.

Transitioning to distributed operations has saved a considerable amount of money during the extended mission. The principal expense change with the implementation of remote operations was the savings on travel expense. Approximately \$250 per day was expensed to the project for a typical visiting scientist on MER to cover travel-related expenses. During the period of remote operations, it is estimated that about 13 outside scientists were necessary to operate each rover – 26 in total. The cost of this team's residence at JPL for operations is estimated at approximately \$200,000 per month of dual-rover operations. The current implementation of remote operations for MER

saves the MER Project at minimum this amount each month.

There are also some modest remote operations related expenses. These expenses included development cost for the remote operations infrastructure and the on-going maintenance of this infrastructure. However, since most of the hardware was existing equipment at JPL that was relocated to remote sites and the maintenance engineers are already on staff, the added cost for remote operations was small relative to the overall travel cost savings.

The MER science team has adapted quite well to remote operations. The team has consistently been able to produce the detailed activity plans and associated sequences on time for each rover on every sol during the period of remote operations. This is likely due to the extensive training the science team received prior to the beginning of rover surface operations - the intense co-located operational period of the prime mission phase (90 sols) which precisely honed each team members operational skills. It is also aided by the similarity of the remote operations infrastructure to the prime mission operational infrastructure (e.g., tools and procedures, etc.).

According to MER Principle Investigator Steve Squyres, there has been no loss in capability of the science team due to moving to the distributed missions architecture. He reports that the system is highly robust – planning at Cornell University continued even during a power failure via a laptop and cellular phone modem. Having the scientists at home with their family and colleagues has been good for morale, and it has enabled the team to successfully retain its members in the long term.[3]

There are still challenges in using the distributed architecture. Communication is more difficult when people are not in the same room. Also, a significant amount of time was spent emailing screen shots back and forth, due to the fact that many mission computer programs were not designed to be collaborative over a distance. Much of the communications difficulties were mitigated by the use of teleconferencing equipment, web cameras, Virtual Network Computing, and SSH. There are aspects to the system that need improved, however, the net impact of moving to a distributed architecture is overwhelmingly positive.

7. RECOMMENDATIONS FOR THE FUTURE

Future missions will be more ambitious and will require significantly more complex supporting technology. The amount of data returned will skyrocket as novel communications infrastructures are utilized. Planning will likely become more specialized and more involved as larger, more sophisticated science payloads are sent to space. Also, the trend towards greater international cooperation in space exploration continues to diversify the geographical distribution of labor.

Consequently, future missions are likely to be partially or fully distributed in their ground support architecture. This requires a new breed of operations software – software designed from the ground up to support collaboration from multiple locations. This software will have to deal with issues of versioning, concurrency, data propagation, content synchronization, and effective communications among team members. To support its development, we make the following recommendations:

- 1) **Design operations software as fundamentally distributed – from the beginning.** The single most challenging aspect of the MER distributed operations effort was coercing centralized software into functioning properly in a distributed environment. The performance of the system was adversely affected by many compromises made to meet the requirements of the original software, and potential robustness was sacrificed due to the need of ad-hoc solutions.
- 2) **Avoid polling by using a common event notification system.** The largest obstacle to good performance in our system was the lack of a common event notification system. Almost all processes were forced to rely on polling, which was inefficient, high in latency, and consumed enormous system resources. It was also a source of unnecessary complexity. A common event notification system would solve this problem by making processes aware of new or newly changed resources. The File Exchange Interface includes a subset of this desired functionality. This is the most significant feature that the current architecture lacks.
- 3) **Distributed planning software should operate in a high-concurrency environment.** Many parts of the planning process are by nature collaborative, as is the scientific process itself. Plan sharing in Distributed SAP was problematic because it was not designed to be highly concurrent. Systems should be designed to allow multiple users to edit a resource simultaneously, or prevent resource conflicts. This might be best accomplished using notions of resource ownership, or a “copy on write” model. Operations should be transactional and reversible where possible.
- 4) **Minimize latency effects in collaborative functionality.** The latency involved in the shared planning subsystem, while low, is still high enough to discourage its use in the interactive part of a planning session. It is important to use techniques, such as optimistic locking in order to reduce the inevitable effects of latency in this kind of global system.
- 5) **Utilize decentralized architectures for information propagation where possible.** As the volume of mission information increases, it will become more critical to use a less centralized method for distributing static information (data products). We suggest the use of swarming downloads (as demonstrated by the BitTorrent protocol) or localized synchronization of caches (swarm caching). The ability for operations

software to obtain data from a localized peer should increase effective throughput significantly, and also lower latency. The network topology in this operations model consists of small, interconnected "clusters," which is ideal for swarming transfers.

- 6) **Use SQL databases where possible.** SQL databases are an industry standard, and they have successfully solved many problems relating to concurrency, locking, transactions, data propagation, and querying. The use of SQL databases for targets was critical for the success of Distributed SAP. To aid in the use of SQL, object oriented software can use what is known as an Object-Relational Bridge (ORB), a piece of software that converts object graphs in memory to a database representation, and back. Distributed SAP uses Castor Java Data Objects to perform this function.
- 7) **Use XML or ASCII Text for metadata.** Lack of integration with specialized science analysis and planning software lead to extra work and reduced capabilities for MER scientists. The use of platform/language neutral standard formats would greatly increase the interoperability of mission software.

8. CONCLUSION

The Distributed Science Activity Planner has contributed to the success of distributed operation for the Mars Exploration Rover mission. Scientists were able to analyze data and plan from their home institution, and collaborate with other scientists around the world. The distributed operations architecture has enabled a large science team to operate Spirit and Opportunity well beyond the original mission lifetime as they continue to return valuable scientific information to earth. Distributed MER operations will serve as a model for missions into the future.

REFERENCES

- [1] Robert Steinke, Paul G. Backes, and Jeffrey S. Norris. Distributed mission operations with the multi-mission encrypted communication system. In *Proceedings IEEE Aerospace Conference*, Big Sky, Montana, March 9-16 2002.
- [2] Paul G. Backes , Jeffrey S. Norris , Mark W. Powell , Marsette A. Vona , Robert Steinke , and Justin Wick. The Science Activity Planner for the Mars Exploration Rover Mission: FIDO Field Test Results. In *Proceedings IEEE Aerospace Conference*, Big Sky, Montana, March 8-15 2003.
- [3] Personal correspondence with Professor Steve Squyres, December 8th, 2004.

BIOGRAPHY



Justin Wick worked as a member of the operations staff of the Mars Exploration Rover mission at the Jet Propulsion Laboratory during 2004. He designed and implemented the Distributed Science Activity Planner system to enable collaboration between scientists at remote locations. Prior to the mission Justin spent four years pursuing a bachelors of science from Cornell University in Applied and Engineering Physics. While not working on MER, Justin parallelized several Magnetohydrodynamic simulation codes for theoretical astrophysics, created a kinematics simulation system for the Cornell RoboCup robotic soccer team, and lead the computer science team on the Cornell Snake Arm robotics project.

Justin currently lives in Ithaca, NY, working for Maas Digital, creating custom computer graphics algorithms for high definition IMAX films. He plans to pursue a career in computational physics.



Dr. John L. Callas received his Bachelor's degree in Engineering from Tufts University in 1981 and his Masters and Ph.D. in Physics from Brown University in 1983 and 1987, respectively. After completing his doctorate in elementary particle physics in 1987, he joined the Jet Propulsion Laboratory in Pasadena, California to work on advanced spacecraft propulsion, which included such futuristic concepts as electric, nuclear and antimatter propulsion.

In 1989 he began work supporting the exploration of Mars with the Mars Observer mission and has since worked on seven Mars missions. In 2000, Dr. Callas was asked to join the Mars Exploration Rover (MER) Project as the Science Manager. Dr. Callas continues as the Science Manager for the highly successful Spirit and Opportunity rovers. Recently, Dr. Callas has begun serving as a Mission Manager on MER, as an additional duty, as the rovers continue their great success on the surface of Mars. In addition to his Mars work, Dr. Callas is involved in the development of instrumentation for astrophysics and planetary science, and teaches mathematics at Pasadena City College as an adjunct faculty member. In his spare time, he mentors students interested in science and works with schools classrooms on science projects.



Jeffrey S. Norris is a computer scientist and member of the technical staff of the Mobility Systems Concept Development section at the Jet Propulsion Laboratory. At JPL, his work is focused in the areas of distributed operations for Mars rovers and landers, secure data distribution, and science data visualization.

Currently, he is a software engineer on the Mars Exploration Rover ground data systems and mission operation systems teams. Jeff received his Bachelor's and Master's degrees in Electrical Engineering and Computer Science from MIT. While an undergraduate, he worked at the MIT Media Laboratory on data visualization and media transport protocols. He completed his Master's thesis on face detection and recognition at the MIT Artificial Intelligence Laboratory. He lives with his wife, Kamala, in La Crescenta, California.



Mark W. Powell has been a member of the technical staff in the Mobility Systems Concept Development section at the Jet Propulsion Laboratory, Pasadena, CA, since 2001. He received his B.S.C.S. in 1992, M.S.C.S in 1997, and Ph.D. in Computer Science and Engineering in 2000 from the University of South

Florida, Tampa. His dissertation work was in the area of advanced illumination modeling, color and range image processing applied to robotics and medical imaging. At JPL his area of focus is science data visualization and science planning for telerobotics. He is currently serving as a software and systems engineer, contributing to the development of science planning software for the 2003 Mars Exploration Rover mission and the JPL Mars Technology Program Field Integrated Design and Operations (FIDO) rover task. He, his wife Nina, and daughters Gwendolyn and Jacquelyn live in Tujunga, CA.



Marsette A. Vona, III is currently a PhD candidate in Computer Science at Massachusetts Institute of Technology. He is a former member of the technical staff of the Mobility Systems Concept Development Section at the Jet Propulsion Laboratory. At JPL, his work was focused in the areas of high-performance interactive 3D

data visualization for planetary exploration, user-interface design for science data analysis software, and Java software architecture for large, resource-intensive applications. Marsette received a B.A. in 1999 from Dartmouth College in Computer Science and Engineering, where he developed new types of hardware and algorithms for self-reconfigurable modular robots. He completed his M.S. in 2001 at MIT's Precision Motion Control lab, where he developed a high-resolution

interferometric metrology system for a new type of robotic grinding machine. Marsette was awarded the Computing Research Association Outstanding Undergraduate Award in 1999 for his research in Self-Reconfigurable Robotics.

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. We would like to thank Professor Steve Sqyures, Mary Mulvanerton, Daniel Crotty, Rupert Scammell, and Adam Wick for their generous assistance in the preparation of this paper.

SAP (MER, merb/ops/ops): Uplink Browser

Browser Edit View

- external
- geo
- http
- mb
- mi
- mtes
- pan
- rat
- sol
- sowg
- external
- sol_339_sowg_science_plan-apxs.rml
- sol_339_sowg_science_plan-core.rml
- sol_339_sowg_science_plan-mb.rml
- sol_339_sowg_science_plan-merged.rml
- sol_339_sowg_science_plan-microimager.rml
- sol_339_sowg_science_plan-mites.rml
- sol_339_sowg_science_plan-nav_haz.rml
- sol_339_sowg_science_plan-other.rml
- sol_339_sowg_science_plan-pancam.rml
- sol_339_sowg_science_plan-pan.rml
- working

Observations Targets

Name	Filter	Priority
ENG_Mini-TES_Sky	ENG_Min-TES_Sky_AND_Ground	High
Elevations +30 deg	ENG_Pancam_Tau	High
pointing_checkout	pre_drive_heatsi	High
ENC_Pancam_Tau	4F_caltarget	High
Pre_drive_heatsi	Pre_drive_shoulder	High
PANCAM_SINGLE_POSITION	Bump	High
PANCAM_SINGLE_POSITION	Drive_to_chairside	High
ROVER_DRIVE_BLIND	Turn to Comm	High
PLACEHOLDER_SCI	Penultimate front	High
HAZCAM_FRONT	Final_Ultimate_Fro	High
HAZCAM_FRONT	Final_Ultimate_Re	High
HAZCAM_REAR		High

center_point center pointing of the desired region

Target: **p2**

Frame: RVR_AZ

Az (deg): 0

Ei (deg): 0

Filter: NA

camera_selection array of filters, downsampling, compression, co...
 DSmp CompR 12108 Downlink Priority

Filter	Priority
L1 empty	1 1.6
L2 750	1 8
L3 670	1 1.6
L4 600	1 1.6
L5 530	1 8
L6 480	1 1.67
L7 430	1 1.67
L8 440ND	1 1.6
R1 430	1 1.67
R2 750	1 1.6
R3 800	1 1.6
Pancam_High_4	1 1.6

Observation

Instrument: **cpu**

- SUE_NOTE
- SFOS_NOTE
- AUTO_SHUTDOWN
- AUTO_WAKEUP
- DEEP_SLEEP
- WAKE_FOR_DEEP_SLEEP
- SAPP_DATA_PRODUCTS
- POWER_THERMAL_DATA_PRODUCTS
- DELETE_DATA
- CHANGE_DOWNLINK_PRIORITY

Uplink Palette

New Activity

- ENG_Mini-TES_Sky_AND_Ground
- ENG_Pancam_Tau
- pre-drive_rock
- Drive_to_Seal
- Mid-drive imaging
- duration_placeholder_TBR
- duration_placeholder_TBR
- PANCAM_MOSAIC
- PANCAM_MOSAIC
- PANCAM_SINGLE_POSITION
- PANCAM_SINGLE_POSITION
- PANCAM_SINGLE_POSITION
- PANCAM_SINGLE_POSITION
- duration_placeholder
- duration_placeholder

Details

center pointing of the desired region

Target: **p2**

Frame: RVR_AZ

Az (deg): 0

Ei (deg): 0

Filter: NA

camera_selection array of filters, downsampling, compression, co...
 DSmp CompR 12108 Downlink Priority

Filter	Priority
L1 empty	1 1.6
L2 750	1 8
L3 670	1 1.6
L4 600	1 1.6
L5 530	1 8
L6 480	1 1.67
L7 430	1 1.67
L8 440ND	1 1.6
R1 430	1 1.67
R2 750	1 1.6
R3 800	1 1.6
Pancam_High_4	1 1.6

Pie Chart

Activity	Value
ENG_Mini-TES_Sky_AND_Ground	24.4
pre-drive_rock	9.1
Drive_to_Seal	4.9
Mid-drive imaging	4.9
Drive_to_chairside	2.5
Post_drive_Hazcam_Navcam_3x1	12.6
ODY_Mini-TES_Sky_AND_Ground	2.9
AML_340_Mini-TES_Sky_AND_Ground	4.9
Other	4.9

Figure 2 – Uplink Browser. The Uplink Browser allows users to edit plans. Two plans are currently open, along with a resource analysis graph. The details dialog box contains additional settings for a planned panoramic camera observation. The upper left contains a file tree of available plan files. The lower left contains a list of available activity types.

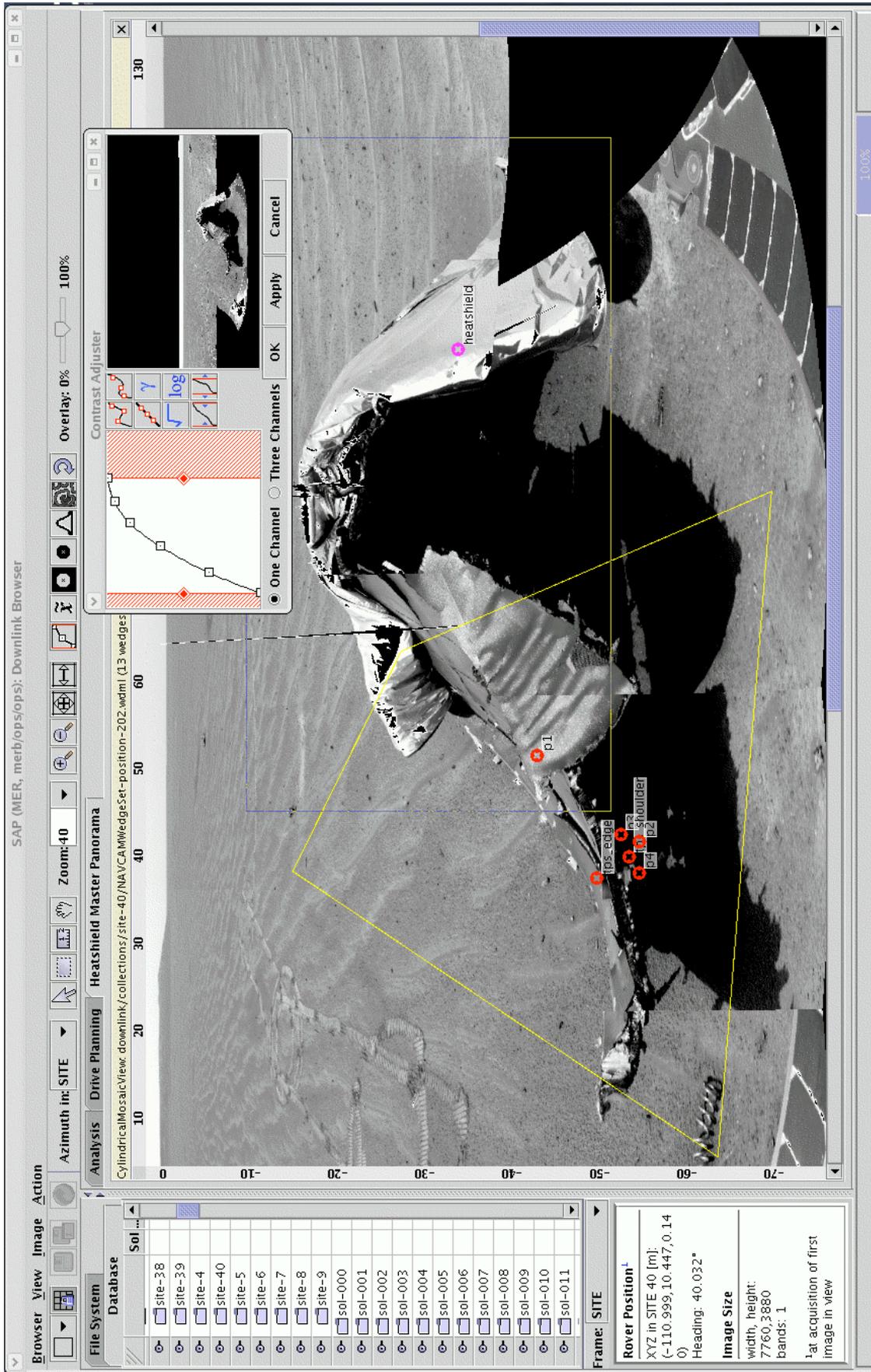


Figure 3 – Downlink Browser. The Downlink Browser allows users to view images and create targets. In this figure, an equicylindrical projection of a navigational camera mosaic is being contrast stretched. Red circles indicate targets, and the yellow trapezoid is the predicted “footprint” of a planned panoramic camera observation. The yellow rectangle is a measuring tool.