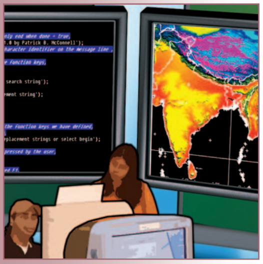


The Collaborative Information Portal and NASA's Mars Rover Mission

NASA Ames Research Center and the Jet Propulsion Laboratory jointly developed the Collaborative Information Portal for NASA's Mars Exploration Rover mission. Mission managers, engineers, scientists, and researchers used this Internet–based enterprise software application to view current staffing and event schedules, download data and image files generated by the rovers, send and receive broadcast messages, and get accurate times in various Mars and Earth time zones. This article describes the application's features, architecture, and implementation, and conveys the lessons learned from its deployment.

Ronald Mak
University of California, Santa Cruz

Joan Walton
NASA Ames Research Center

In January 2004, the Mars exploration rover (MER) mission successfully deployed two robotic geologists — Spirit and Opportunity — to opposite sides of the red planet. As Figure 1 shows, the rovers carry impressive arrays of cameras and scientific instruments that send data and images back to Earth, where ground-based systems process and store the information in file servers.[1] After analyzing the files, NASA scientists have concluded that liquid water did in fact exist on the surface of Mars in the distant past.[2,3]

However, the mission isn't just about rovers and data. On Earth, people manage the mission, send commands to the rovers, and analyze the downloaded information. Computer scientists and software engineers at NASA Ames Research Center worked closely with the mission managers at the Jet Propulsion Laboratory (JPL) to create applications that support the mission. One such application, the Collaborative Information Portal (CIP), helps mission personnel perform their daily tasks, whether they work inside mission control or the science areas at JPL, or in their homes, schools, or offices.

With a three-tiered, service-oriented architecture (SOA) — client, middleware, and data repository — built using Java, industry standards, and commercial software, CIP provides secure access to mission schedules and to data and images downloaded from Mars. Its design satisfied JPL's mission requirements for functionality, scalability, and reliability, and users can run the CIP client tools on Windows, Unix, Linux, and Macintosh platforms.

The CIP Client Application

Mission managers and engineers working inside JPL's mission control use NASA's Deep Space Network (DSN), an international network of antennas, to send command sequences that direct the rovers' operations, receive telemetry information, and download the data and images generated by the rovers' scientific instruments and cameras. Scientists and researchers working at JPL and its partner institutions analyze the downloaded data and images and plan the rovers' operations for the next day. Each rover has its own team supporting it, although some people move between teams. Most people work on Mars time (a Martian day, or *sol*, is roughly 40 minutes longer than an Earth day), but each person can have different roles at different times of the day. Although the mission managers, engineers, scientists, and researchers have different roles and tasks, CIP provides tools and functions that support all sets of users.

Figure 2 shows how the CIP client application assists the rover teams with their daily tasks by consolidating several useful tools into a single consistent and intuitive user interface. CIP shows the current time in Mars and Earth time zones, and it displays both staff and event schedules. Users can browse the data repositories and view the downloaded data and images. They can also send and receive broadcast messages.

Schedule Viewer

Mission personnel often refer to CIP's staff and event schedules, especially if they work on Mars time. The schedule viewer tool displays the durations of events and staff role assignments as horizontal time lines, and it can show multiple time scales for Mars and Earth time zones. Each rover is in a separate Mars time zone, and people working in various countries use different Earth time zones. The tool tells its users when events are due to occur, and it indicates who is working when and where, and what roles they need to fill that day. The schedules help people adhere to Mars time: the extra 40 minutes means that, relative to Earth time, regularly scheduled events drift later from day to day. CIP keeps the schedules in a database in the data repository tier.

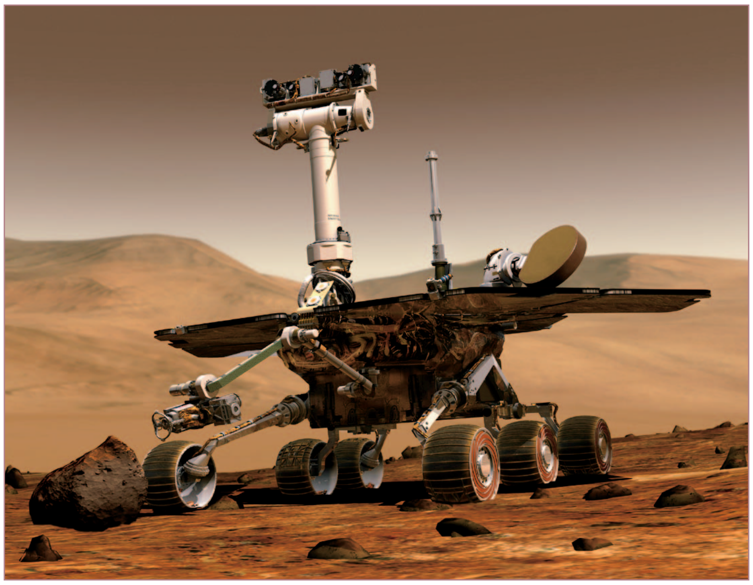
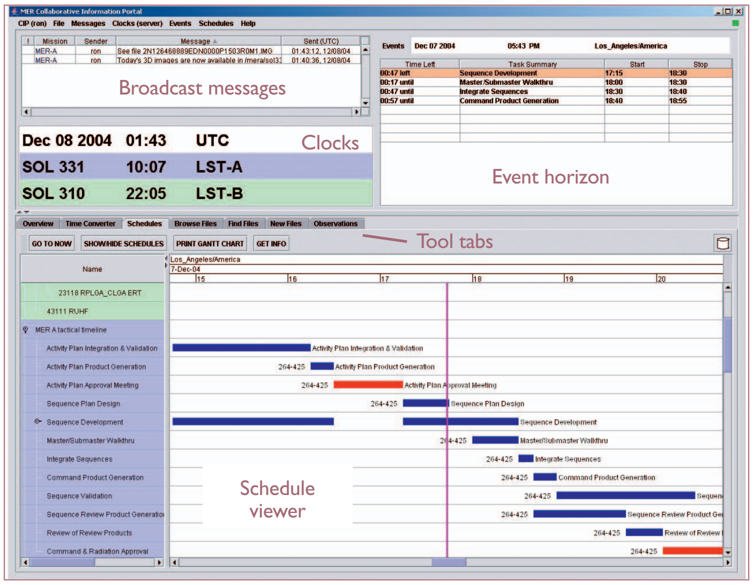


Figure 1. Artist's depiction of a rover working on Mars. Various scientific instruments are deployed on the front articulated arm and on top of the main mast that rises above the solar panels. The rover also carries several cameras. The smaller mast is the low-gain antenna, and the high-gain antenna is the circular disk at the right. (Image courtesy of NASA and JPL.)


MER Collaborative Information Portal
CIP (ron) File Messages Clocks (server) Events Schedules Help
Broadcast messages
Clocks
Event horizon
Overview Time Converter Schedules Browse Files Find Files New Files Observations
GO TO NOW SHOW/HIDE SCHEDULES PRINT GANTT CHART GET INFO
Tool tabs
Schedule viewer

!	Mission	Sender	Message	Sent (UTC)
	MER-A	ron	See file 2N126468889EDN0000P1503R0M1.IMG	01:43:12, 12/08/04
	MER-A	ron	Today's 3D images are now available in /mera/sol33	01:40:36, 12/08/04

Dec 08 2004	01:43	UTC
SOL 331	10:07	LST-A
SOL 310	22:05	LST-B

Events Dec 07 2004 05:43 PM Los_Angeles/America

Time Left	Task Summary	Start	Stop
00:47 left	Sequence Development	17:15	18:30
00:17 until	Master/Submaster Walkthru	18:00	18:30
00:47 until	Integrate Sequences	18:30	18:40
00:57 until	Command Product Generation	18:40	18:55

Los_Angeles/America
7-Dec-04
15 16 17 18 19 20

Name	
23118 RPLGA_CLGA ERT	
43111 RUHF	
MER A tactical timeline	
Activity Plan Integration & Validation	Activity Plan Integration & Validation
Activity Plan Product Generation	264-425 Activity Plan Product Generation
Activity Plan Approval Meeting	264-425 Activity Plan Approval Meeting
Sequence Plan Design	264-425 Sequence Plan Design
Sequence Development	Sequence Development
Master/Submaster Walkthru	264-425 Master/Submaster Walkthru
Integrate Sequences	264-425 Integrate Sequences
Command Product Generation	264-425 Command Product Generation
Sequence Validation	264-425 Sequen
Sequence Review Product Generatio	264-425 Sequence Review Product Ge
Review of Review Products	264-425 Review of Review
Command & Radiation Approval	264-425

Figure 2. A screen shot of the CIP client application interface. The broadcast announcements, clock, event horizon, and schedule viewer tools are currently visible. The tool tabs provide access to the time converter and data navigator tools.

Event Horizon

Users can place scheduled events into the event horizon tool, which then displays a running countdown of the time left until the start of the event. Event table rows change colors to warn of impending start times. This tool is useful for all mission personnel; it helps ensure that people show up to meetings on time, or that they're prepared for important events such as the next command sequence uplink to a rover.

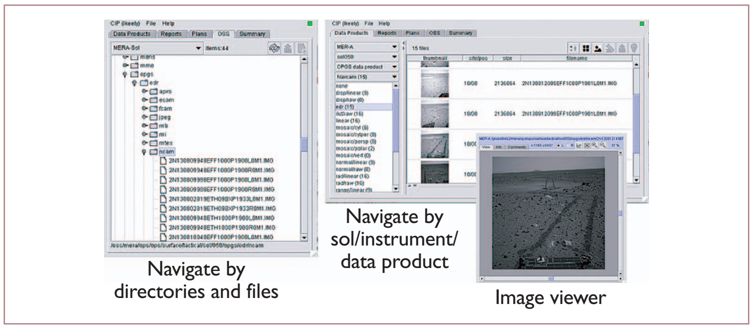

Navigate by directories and files
Navigate by sol/instrument/ data product
Image viewer

Figure 3. Screen shots of the various data navigator tools. Users can navigate the mission file servers via the Unix file directory structure, or they can browse the data products organized by sol and rover instrument. Various report and image viewers display file contents. (Images courtesy of NASA and JPL.)

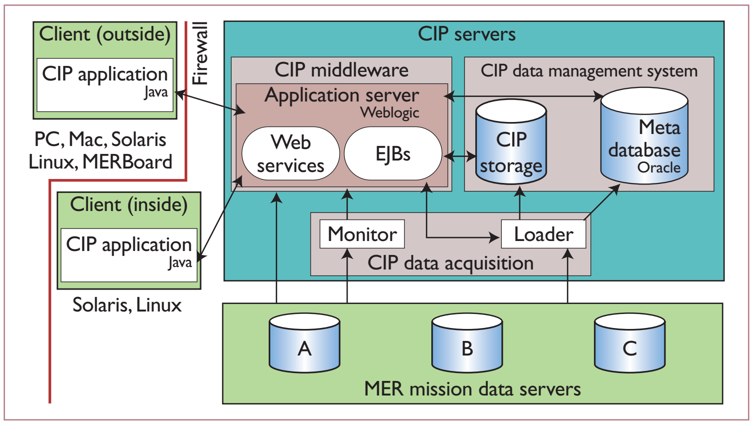


Figure 4. Overview of the CIP architecture. The service-oriented architecture is based on Web services and J2EE, and consists of client, middleware, and data repository tiers. The middleware, database, and file servers are located securely behind the mission firewall. Clients access the CIP middleware over the Internet via Web services.

Data Navigator

The DSN sends the downloaded data and images to JPL, which stores them in mission file servers under a hierarchical Unix directory structure. Mission engineers can do further processing of the data and image files, also known as data products, such as image color correction and assembling three-dimensional pictures.

NASA scientists and researchers use CIP's data navigator tools to access the mission file servers. CIP transports data and images securely over the Internet using Web services and the HTTPS protocol. As Figure 3 shows, one tool enables users to navigate the Unix directories, and another tool organizes the data products by rover, sol, and instrument for users to browse and access.

CIP's data repository tier generates metadata for the data products, which it stores in a database. Example metadata fields include which rover and which instrument or camera created the product and during which sol. Based on this metadata, the data navigator tools automatically classify and organize the data products. Whenever a user wants to view a data product, the classification enables CIP to determine which viewer to bring up. Still other data navigator tools let users search for data products according to metadata field values.

Clocks

"What time is it?" is a key question for everyone working on the MER mission. Not only does the mission run on Mars time, but the teams work in two Martian time zones and various Earth time zones. To contend with this issue, the CIP client application displays clocks that show Mars and Earth times in user-chosen time zones.

Broadcast Announcements

The broadcast announcements tool lets CIP users send messages to other users — typically, announcements about the availability of new data products such as a newly created three-dimensional image. CIP archives all the broadcast messages into a database in the data repository tier, which users can browse to review past messages and retrieve messages they might have missed while they were away from CIP.

CIP Architecture

Figure 4 shows CIP's SOA. Partitioning the software into client, middleware, and data repository tiers balances and distributes the computational resources, and enables CIP to meet its goals of scalability, reliability, extensibility, and security. Its middleware automatically replicates and pools its service components to handle peak loads, and these components have automatic error recovery to enhance overall system reliability. The architecture is based on components that are straightforward to replace or add new ones to. CIP provides security at various levels, such as encrypted data transmissions and user authentication and authorization.

Client Tier

Users run the Java-based CIP client application on both their desktop and laptop computers. We designed CIP to be a thick-client application that can run by itself, as opposed to a thin-client application that must run within a Web browser. The client contacts the middleware over the Internet only when it needs to request a service, such as when a user clicks a button. It polls the middleware periodically for the current time and for any new broadcast messages. We implemented the client application using the Java platform with graphical user interface components from its Java Foundation Classes ("Swing").

Figure 5 shows our component-based approach for the client tier. Each client tool is a CIP *component object*. A *service manager* object supports one or more component objects, and it makes calls into the Web services client stub that connects to a particular remote middleware service — the clock components, for example, use the time service manager object, whose client stub connects to the middleware's time service.

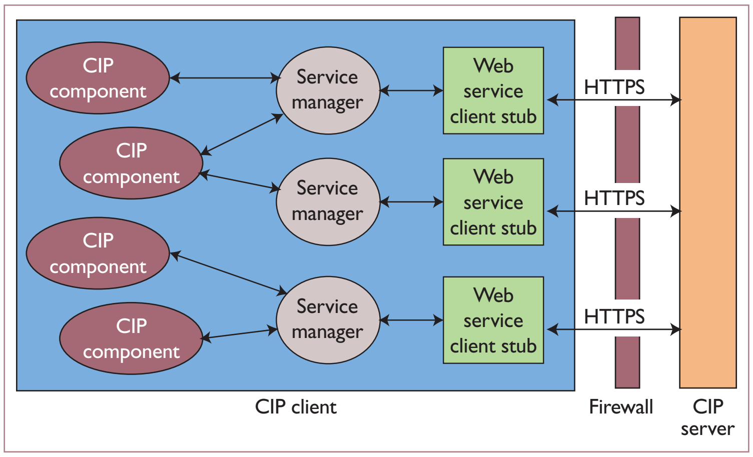


Figure 5. CIP client application's component-based architecture. A service manager object supports one or more client component objects, and it makes calls into the Web services client stub that connects to a particular remote middleware service.

Middleware Tier

The CIP middleware runs on a server at JPL and communicates over the Internet with all actively running client applications. The clients may simultaneously request services, so the middleware handles the load by replicating, pooling, and caching its service components.

We designed the middleware using SOA principles and two industry standards: Java 2 Enterprise Edition (J2EE; http://java.sun.com/j2ee/) and Web services (www.w3.org/2002/ws/). At runtime, a commercial off-the-shelf application server, WebLogic from BEA Systems (www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/server/), manages the J2EE-based components that we developed called Enterprise JavaBeans (EJB).

The client applications request services from the middleware, which returns a response to each request. These services include

- the *user management service* to process user logins and logouts and to maintain user sessions;
- the *time service* to provide Mars and Earth times in various time zones;
- the *metadata query service* to fetch metadata from the database;
- the *schedule query service* to fetch schedules from the database;
- the *file streamer service* to download and upload files; and
- the *message service* for asynchronous notification and broadcast messages.

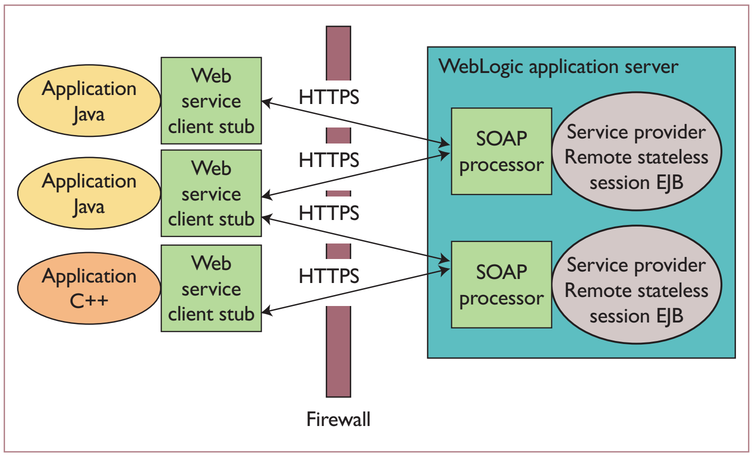


Figure 6. Web services and the service provider EJBs. A service provider, implemented as a stateless session bean, represents each middleware service. To handle heavy service request loads, the application server replicates and pools instances of the service provider bean. For security, all SOAP transmissions are encrypted with the HTTPS protocol.

Figure 6 shows how one or more service provider EJBs, which are stateless session beans, represent each service. Each bean has public methods, or procedures, that client applications invoke remotely over the Internet to request services. The application server maintains an instance pool of these stateless beans; specifically, it creates or destroys these instances in response to the request load, which makes CIP scalable. As users make more requests, the application server automatically replicates more service providers to handle them.

Several of the middleware services create data beans, which are stateful session EJBs. Because these beans maintain state information, the application server caches them in memory. The metadata and schedule query services, for example, create data beans that use Java Database Connectivity (JDBC) calls to query the database repositories (http://java.sun.com/products/jdbc/). Each data bean keeps a reference to the returned query results, so by taking advantage of the application server's memory cache, the query services greatly improve their performance for repeated requests for the same data. If the data beans are already in the cache, the service doesn't need to make the more time-consuming database queries.

A key middleware innovation is our use of Web services. As shown in Figures 5 and 6, client applications use Web services to communicate with the remote service provider EJBs. Each client service manager object has a Web services client stub that acts as a proxy for the remote service provider bean.

We developed each service as a collection of public methods that a client application can invoke remotely. When a client application requests the current Mars time, for example, the time service manager makes a local call to the client stub's *getMarsTime* method. The stub converts the call to a service request in the form of a text document encoded in the XML-based SOAP protocol, as defined by the Web services standard. The stub then sends the document over the Internet to the middleware's time service using the encrypted HTTPS protocol. The time service's SOAP processor decrypts the request and invokes the time service provider bean's public *getMarsTime* method. The response returns similarly across the Internet to the requesting client application as an encrypted SOAP document. The client stub then decrypts the response and converts it to Java objects for the client application.

Therefore, client applications make local method calls to client stubs and get results back from them. Web services handle all the details for connecting to the remote middleware service, encryption and decryption, and requests and responses sent across the Internet.

Organizing the middleware as a collection of loosely coupled services makes CIP extremely extensible: it is easy to plug and play new services and replace or remove obsolete ones. The application server enhances reliability by monitoring service operation and automatically doing any necessary retries and error recoveries.

The middleware logs every activity, including user requests. For each request, the log entry contains a timestamp, the username, the name of the called method, details about the request, and key information about the results. We can do data mining in these logs afterwards to compute various statistics, such as how frequently users accessed certain types of schedules, and to deduce usage patterns, such as how users search for data products. This helps us fine-tune the middleware's operations.

We developed a separate client-side utility program to constantly monitor the middleware's status and to report statistics, such as memory usage and response times, graphically. Knowing the server's health at all times enables the system operators to correct problems before they became serious.

Before deploying the CIP middleware for operation, we put it through intensive stress testing. We developed a stand-alone interactive utility to perform the stress testing by simulating any number of users performing various client functions, such as accessing schedules or downloading files. This testing pointed out performance bottlenecks and helped us determine the load CIP could handle. We can increase the load capacity by adding more server hardware, but throughput is limited by network latencies.

Dynamic reconfiguration allows CIP to stay up and running for long periods (more than 77 days at a time) before scheduled server maintenance shutdowns. We designed the individual middleware services to be hot redeployable, meaning we can restart a service while the rest of the middleware (and CIP as a whole) continues to run. To reconfigure a service, a system administrator first edits the service's configuration file (to change the one-way light time, for example, which is the time it takes a signal to travel from Earth to Mars) and then redeploys the service. When the service restarts, it reads in its new configuration. Redeploying a service typically takes a few seconds, and users rarely notice any interruptions.

CIP security is a combination of user management and data encryption. The CIP middleware requires each user to log in with a username and a password. Each user has pre-assigned privileges that grants or forbids access to certain data or images. Digital certificates from Verisign help encrypt all data traffic between the middleware and the users' client applications (www.verisign.com).

Asynchronous Messaging

CIP has two types of asynchronous messages:

- CIP's data repository tier sends *notification messages* to the middleware and to users whenever new data and image files are available.

- Users can send a *broadcast messages* to other CIP users to inform them, for example, of a just-completed analysis of an image file.

To implement asynchronous messages, the CIP middleware uses the Java Message Service (JMS; http://java.sun.com/products/jms/), which follows a publish-subscribe model. JMS uses objects called *topics* to represent different types of messages; message consumers, such as a CIP client application, can subscribe to (register interest in) one or more topics. Whenever a message producer, such as another CIP client application or some other CIP component, publishes (sends) a message to that topic, JMS delivers the message to all the message consumers that subscribe to that topic.

Users interested in panoramic camera images, for example, subscribe to the pancam images topic, and whenever the data repository tier detects a new panoramic camera image, it publishes a message to that topic. The interested users receive the messages via Web services the next time their client applications poll the middleware for messages. CIP messaging is asynchronous, meaning message queuing and delivery occur in parallel with all other operations.

A message-driven EJB subscribes to the broadcast messages topic. It receives and archives all the broadcast messages in a database so that users can browse the archived messages to review them and to pick up any missed messages.

Data Repository Tier

Figure 7 illustrates the data repository tier, which encompasses CIP's Oracle databases and the MER mission file system, both of which run on separate servers at JPL.

The file monitor constantly watches the logs generated by the Unix utility program *nfslogd*, which writes a log entry every time a file is created, read, moved, or updated (see http://mirrors.ccs.neu.edu/cgi-bin/unixhelp/man-cgi?nfslogd+1/). The file monitor has a configuration file with regular expressions representing the file paths relevant to CIP, and it filters out any files whose paths don't match these expressions.

Unlike the file monitor, the file detector uses the Unix utility program *find* to walk the mission file system's directory tree in search of any relevant newly created or updated files (see http://mirrors.ccs.neu.edu/cgi-bin/unixhelp/man-cgi?find+1). The file detector walks the directories once during each run and acts as a backup for the file monitor whenever *nfslogd* isn't running.

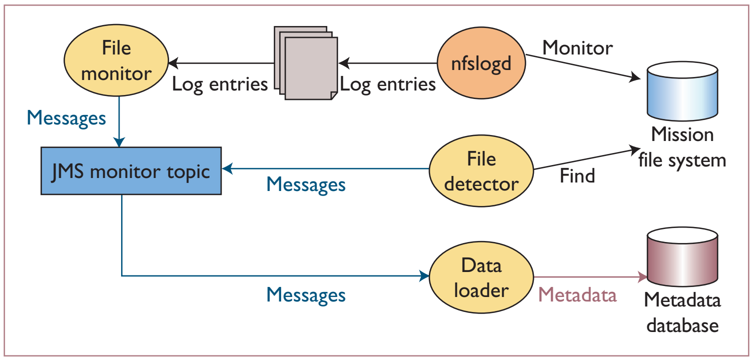


Figure 7. The data repository tier. The file monitor or detector monitors the mission file servers. When either one detects a newly created or updated file, it sends a message to the data loader, which then generates metadata for that file.

As soon as either file monitor or file detector encounters a relevant new or updated file, it sends a message to the data loader, which generates metadata for that file. Using regular expressions from a configuration file, the loader derives metadata field values from the file path itself. The loader also obtains information from the Unix file system; for some types of files, it reads the file header to get additional metadata field values. Example metadata fields include the file name, its creation date and time, the rover the file belongs to, the rover's location, which rover instrument generated the file data, during which sol, and so on. The loader then inserts or updates the metadata in the database.

Following industry standards and using proven commercial software for the infrastructure reasonably assured us that the CIP's underlying "plumbing" would work. The real challenges of enterprise development aren't in the coding, but in integrating the various components.[4]

Ever-changing requirements before deployment coupled with ever-changing operational parameters after deployment made it crucial to develop services that were plug-and-play, mutually independent, and dynamically reconfigurable. We also found that both user and stress testing were critical. JPL ran a series of operational readiness tests in which teams of mission managers, engineers, and scientists tested software systems such as CIP under realistic conditions. We found and fixed many bugs during these tests and gained invaluable user feedback. Our stress testing helped us discover the limits of our software before the users did. Real-time server monitoring and logging helped the system operators keep track of what was going on and headed off any potential problems. The middleware logs provided ways to analyze usage patterns and fine-tune the middleware.

We were initially concerned that Web services would cause performance problems, because XML documents for service requests and responses involved so much data conversion, encryption, and decryption. Yet, CIP achieves a data throughput rate of 100-Mbytes per hour between a client application and the middleware, which is usually sufficient.

The Mars rovers continue to operate way beyond their original 90-sol mission; through over 300 sols, CIP has maintained a 99.9 percent uptime record.

CIP's success with the MER mission encourages us to make improvements to the software and add new features for user collaboration and interapplication data exchange. NASA Ames and JPL are currently designing a new version of CIP for possible use by later missions, such as the Mars Phoenix mission that will launch in August 2007 and land the following May (see http://mars.jpl.nasa.gov/missions/future/phoenix.html).

NASA computer scientists are currently investigating ways to define common enterprise software that multiple NASA centers can use for future missions. SOA and other industry standards will play important roles in creating a multicenter, multimission infrastructure that will let reusable software components from different missions communicate with each other and exchange data. Results from these investigations will become increasingly evident in upcoming NASA missions.

Acknowledgments

Funding for CIP's development came from the Computing, Networking, and Information Systems Project and the Intelligent Systems Project within NASA's Computing, Information, and Communications Technology Program. In addition to the authors, other CIP project members included Roy Britten, Louise Chan, Sanjay Desai, Matt D'Ortenzio, Glen Elliot, Robert Filman, Dennis Heher, Kim Hubbard, Sandra Johan, Leslie Keely, Carson Little, Quit Nguyen, Tarang Patel, John Schreiner, Jeff Shapiro, Elias Sinderson, and Robert Wing. This project would not have been possible without the support, assistance, and collaboration of JPL and the MER team.

References

1. "Mars Exploration Rover," NASA fact sheet, Jet Propulsion Laboratory, Oct. 2004; www.jpl.nasa.gov/news/fact_sheets/mars03rovers.pdf.
2. "Opportunity Rover Finds Strong Evidence Meridiani Planum Was Wet," NASA press release, 2 Mar. 2004; http://marsrovers.jpl.nasa.gov/newsroom/pressreleases/20040302a.html.
3. "Spirit Finds Multi-Layer Hints of Past Water at Mars' Gusev Site," NASA press release, 1 Apr. 2004; http://marsrovers.jpl.nasa.gov/newsroom/pressreleases/20040401a.html.
4. R. Mak, "Enterprise Development for Mars and other Alien Places," keynote address presented at BEA eWorld 2004 Conference, May 2004; www.apropos-logic.com/BEA_eWorld_keynote_address.pdf.

Ronald Mak is a project scientist in the University Affiliated Research Center (UARC), which is a partnership between the University of California at Santa Cruz and the NASA Ames Research Center in Moffett Field. He worked on the CIP development team as the architect and lead developer of its middleware. Prior to working at NASA Ames, Mak had more than 15 years of industry experience developing enterprise software systems. He has a BS in the mathematical sciences and an MS in computer science from Stanford University. Contact him at rmak@mail.arc.nasa.gov or at ron@apropos-logic.com.

Joan Walton is group leader of the Information Design Group in the Computational Sciences Division at the NASA Ames Research Center. In her role as a CIP project manager, she led a team of 12 software developers and was responsible for guiding the project's technical direction, tracking its schedule, meeting milestones, and interacting with MER mission management to develop requirements and meet JPL deployment criteria. Walton has a BA in physics from Swarthmore College and an MS in medical information sciences from Stanford University. Contact her at jdwalton@mail.arc.nasa.gov.