AIAA 2001-4681

LEVERAGING MIDDLEWARE-BASED INFRASTRUCTURE FOR REMOTE EXPLORATION

Anthony.Barrett@jpl.nasa.gov, M/S 126-347, 818-393-5372
Thomas.McVittie@jpl.nasa.gov, M/S 126-255, 818-393-5052
Norman.Lamarra@jpl.nasa.gov, M/S 126-201, 818-393-1561
Larry.Bergman@jpl.nasa.gov, M/S 126-254, 818-393-5314
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr, Pasadena, CA 91109

ABSTRACT[*]

Middleware can improve the capability of business and science applications by providing “standard” shared services to reduce the complexity or increase the capability of every participating application. Successful examples of this approach (such as multi-tier client/server and more recent portal-based architectures) have fueled the growth of "enterprise-level" applications, providing better integration and more rapid adaptability of business in many fields. Unfortunately, science and engineering application development has not kept pace with evolving middleware techniques, especially in aerospace and defense systems, partly due to the complexity, criticality, and length of the system lifecycle for such systems (typically many years). We are therefore attempting to reap some of the benefits of middleware-based application development for Remote Exploration, by proposing development of evolvable services to enable the building of enhanced mission applications more simply. This paper describes middleware systems developed at JPL, shows how leveraging middleware implementation strategies can facilitate building a mission operations system for managing multiple interacting missions on Mars, and proposes an approach to implementing demonstrations as part of a roadmap providing progressively more intelligent remote exploration.

1. INTRODUCTION

Over the past forty years fifteen missions have successfully sent spacecraft to Mars. While Mariner 4 was the first spacecraft and simply flew by Mars in 1964, later spacecraft were progressively more complex and capable. Orbiters (like Mariner 9 and Mars 3) followed fly-bys in 1971. While Mars 3 also dropped a lander, it stopped transmitting twenty seconds after landing. Viking 1 heralded the age of Mars landers by reaching the surface in 1976 and surviving for six years. In 1993 the loss of the Mars Observer orbiter signaled the end of billion-dollar ten-year spacecraft development cycles; the focus shifted to using recent miniaturization techniques to develop smaller but more frequent missions and the goal of “faster, better, cheaper” was articulated by NASA. Finally, Pathfinder started the age of robotic rovers in 1997.

As depicted in figure 1, the operations process for past Mars missions, as well as any spacecraft in deep space, involves a six-step cycle including: (1) science plan generation, (2) command sequence generation/ validation, (3) uplink of the new sequence, (4) sensor data acquisition, (5) telemetry downlink, (6) science and engineering analysis. During downlink, a spacecraft transmits mainly raw sensor data to the ground. Engineering analysis extracts spacecraft health and status, while science analysis computes science products that help researchers answer the questions that originally motivated the mission. Often these products raise more questions than they answer, and new investigations to answer these additional questions are inserted into the science plan generation process. The entire set of investigations combine to generate observation schedules, which are passed to the command sequence generation and validation process. This process expands a schedule into a sequence of commands to articulate the instruments and collect data for downlink to Earth, while avoiding any flight rule violations that could endanger the spacecraft. After uplinking the validated sequence, the cycle returns another batch of downlinked data/telemetry to start the next cycle.

Today, we are receiving Mars Global Surveyor telemetry, and several Mars missions are planned over the next several years. In addition to illustrating the six-step operations cycle, figure 1 depicts a view of the “problem space” for communication and control

[*] Copyright © 2001 by the American Institute of Aeronautics and Astronautics, Inc. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

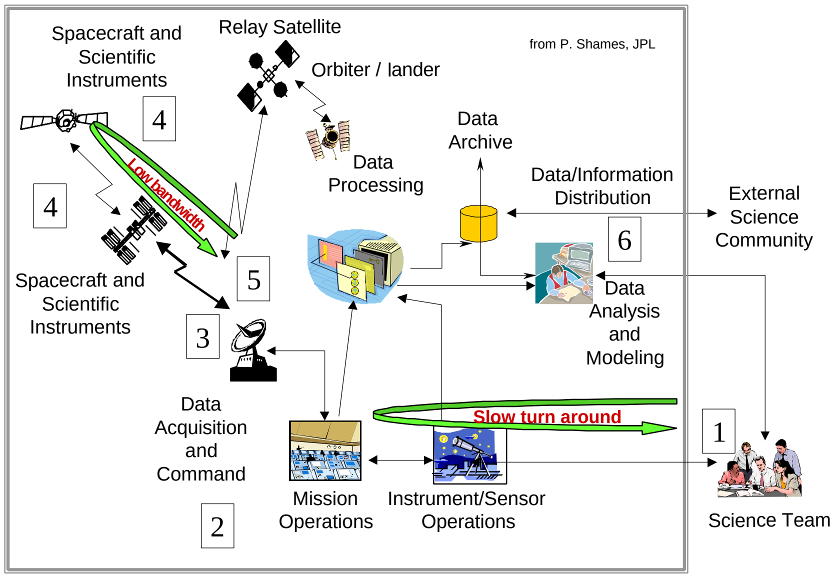
Spacecraft and Scientific Instruments
Relay Satellite
from P. Shames, JPL
Orbiter / lander
4
Data Archive
Data Processing
Low bandwidth
4
Data/Information Distribution
External Science Community
6
5
Spacecraft and Scientific Instruments
Data Analysis and Modeling
3
Data Acquisition and Command
Slow turn around
1
Mission Operations
Instrument/Sensor Operations
2
Science Team

Figure 1. Problem Space Domain for Space Standards Working Group of OMG (see text)

of spacecraft and instruments in deep space, via JPL's Deep Space Network (DSN), showing ground processing, transmission/reception, on-board processing, and eventual science production. While all missions tend to take the same approach to operations, most missions have unique software infrastructures and communications protocols. International bodies such as the CCSDS attempt to standardize such communications protocols, and the newly-formed Space Standards Working Group of the Object Management Group (OMG) is addressing software infrastructure issues. The objectives of these groups mainly revolve around reducing the cost/risk of building mission operations systems.

The main bottlenecks in the current approach to mission operations involve (1) the low communications bandwidth for downlinking data, and (2) the needed negotiations between the science team and the operations staff needed for generating each command sequence. These two bottlenecks drive missions to produce sub-optimal command sequences that produce sharply restricted amounts of data. This paper addresses how standardized middleware software technology can assist in relaxing these bottlenecks for a Mars campaign, and suggests areas for potentially fruitful future study. In the next section we describe 3 approaches to easing the bottlenecks. Section 3 subsequently describes the current state of the art in application middleware in order to set the stage for section 4, where we describe both an existing middleware application and an example scenario on using middleware to enhance exploration on Mars in 2007.

2. IMPROVING ACCESS TO MARS

While more recent Mars missions are more complex, all missions have shared a simplifying assumption – each operated in isolation. By contrast, cooperative missions will succeed such isolated missions in the future, requiring yet further software complexity. The international science community is planning sixteen missions to Mars over the next ten years (figure 2), and these missions will cooperate in multiple ways. Earlier missions will provide precision approach navigation for later missions, and real-time tracking for critical events like descent and landing or orbit insertion. Orbiters will provide relay services to landed assets and positioning services to rovers and other mobile "scout" missions. All missions will cooperate on radiometric experiments and maintaining a common time reference for relating data between missions. These features have been conceptualized as a "Mars Network" of orbiting satellites[1]. While all missions will improve the potential for collecting data on Mars by placing multiple sensors, actually realizing this potential requires improving mission operations tools and techniques both to command the sensors and to collect the resultant raw measurements.

Scientist to Instrument Connectivity

Within the Mars exploration context, one useful operations improvement is to make it easier for scientists to control their experiments. During Sojourner operations[2], new command sequences were uplinked early to mid-morning on Mars and data was

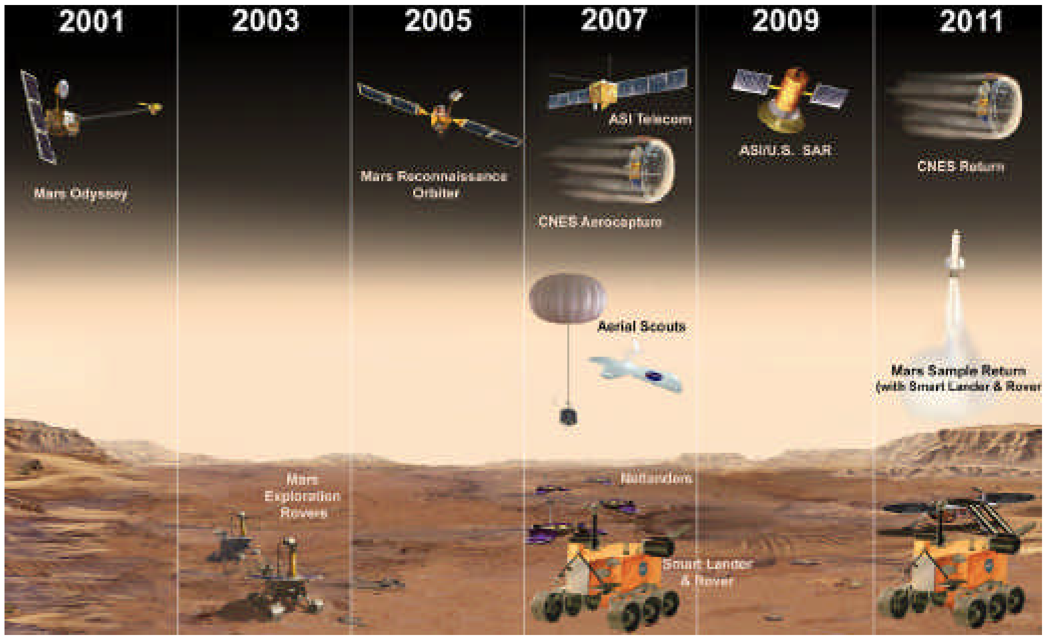

Figure 2. Future Missions to Mars

downlinked at three times during a sol (Martian day): just prior to the uplink, at Martian noon, and mid- to late-afternoon on Mars. While the early downlink was used to assure that no contingencies invalidated the next command sequence, the operations cycle for generating the next day's sequence started after the late-afternoon downlink. Given that Sojourner command sequences were manually generated, the daily operations efforts were arduous. Scientists had to quickly generate an initial observation plan and negotiate with the command sequencing personnel to generate a sequence to uplink. This negotiation can both add and remove observations as opportunities and problems are found while validating the sequence, but the cycle must complete before the next morning's uplink. While this hard deadline was adequate for Sojourner (which could only travel a couple meters per sol), such a deadline would be unnecessarily restrictive for the Mars Exploration Rovers (MER) in 2003. These rovers can travel up to 100 meters per sol, providing many more observation opportunities to choose from on each operations cycle.

One way to accelerate this process (and increase the amount of resultant science data) uses automation both to interpret the downlinked data and to automate parts of the command sequencing and validation steps. In this way, a scientist has more time to generate an observation plan and she can continually validate observation plans during the generation process. Figure 3 illustrates such a system[3] that was prototyped using the ASPEN planner/scheduler and the Web Interface for TeleScience (WITS). While the ASPEN system automated much of the command sequence generation and validation process, WITS provided an interface for developing science plans, resulting in a more direct link between the scientist and her experiment.

Improved Average Downlinked Data Quality

Another approach to improve remote asset utilization involves increasing the amount of *information* delivered to a scientist. While an instrument can produce prodigious amounts of raw data, the communications system can currently only deliver a small percentage of it to a mission scientist. For instance, there are plans to put a camera on the Mars Reconnaissance Orbiter that can image spots on the Martian surface to a 20-cm resolution. At this resolution, there are 3.6×10^{15} pixels on the surface of Mars. Supposing 12 bits of information per pixel, and a 2:1 lossless compression algorithm, the camera can generate 2.2×10^{16} bits of information. Using the DSN, an orbiter like Mars Global Surveyor (MGS) can downlink about 85 Kbps during about 8 track hours per day, resulting in 8.5×10^{11} downlinked bits per Earth year. Thus only 0.0039% of the Martian surface can be delivered per Earth year as raw 20-cm resolution imagery using such a link.

This provides a strong motivation for improving the proportion of desired information in the downlinked data, via intelligent remote data handling. Such intelligence might include techniques for remote science analysis and data mining, data compression and fusion, and event-driven data handling[4]. From

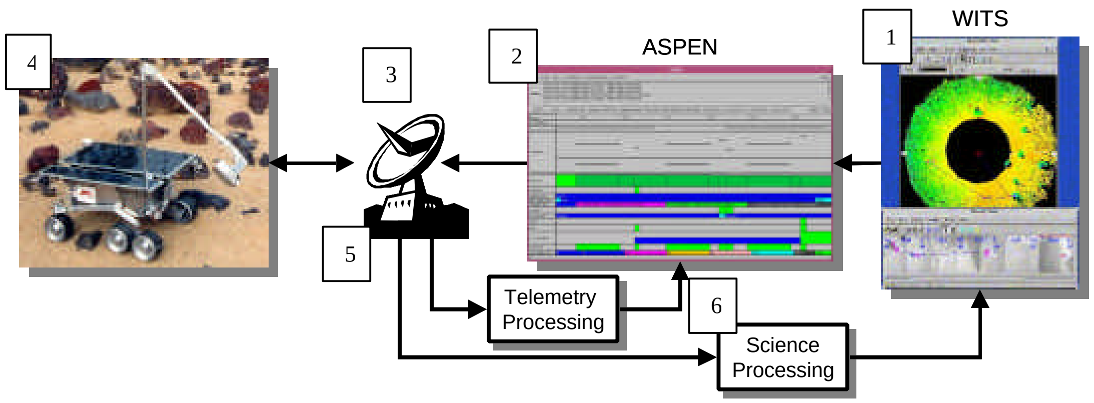

Figure 3. Automated rover command generation with ASPEN and WITS

the perspective of figure 3, we can implement the enhancement by migrating parts of the science-processing task onto the spacecraft in order to prioritize the raw data by its information content. For instance, a scientist might be particularly interested in smooth round Martian boulders and their surrounding environment to prove the persistent existence of running water over Martian centuries. With this in mind, the scientist could utilize an automated on-board pattern-matching algorithm for finding pixel patterns indicating smooth round rocks. She can then request many more desirable images, and use the algorithm to prioritize those for downlink, perhaps dropping low-priority images due to the bandwidth limitations.

Opportunistic Science

A third technique to improve Mars access involves opportunistically gathering science data while interacting with a poorly-modeled environment. For instance, during each sol, MER rovers can travel up to 100m, thus there are high probabilities of unexpected observation opportunities between downlinks. Capturing these opportunities involves extending the remote science processing to initiate new observations as well as prioritize collected data from the observations scheduled from Earth.

Thus, in addition to extending the remote science-processing element, enabling opportunistic science involves migrating parts of all six operations steps onto the spacecraft to enable autonomous operations. With sufficiently successful autonomy, scientists and operations staffs would no longer need to generate a low-level command sequence – rather, they would interact with the spacecraft by sending observation and maintenance goals or agendas. Given these agendas, an on-board science-processing component analyzes sensor data to determine data downlink priorities, makes improvements to an investigation's current observation agenda, and both detects and responds to unexpected phenomena. Observation agendas would include both scheduled periodic observations as well as aperiodic observations in response to detecting unexpected phenomena, and scientists alter these agendas as new discoveries evolve the focus of their investigations. Different agenda components would interact, e.g., the maintenance agenda might affect the telemetry-processing agenda due to detecting wear trends and diagnosing faults.

3. MIDDLEWARE TECHNOLOGY

Three approaches were introduced in the previous section to improve remote exploration: a) improving scientists' control of experiments; b) improving the quality of the downlinked raw data; and c) enabling opportunistic science. This section explores how middleware could assist with addressing each of these needs. The first approach could be addressed by considering the two-way conversation between the scientist and the instrument (figure 1) as utilizing an intelligent "channel", capable of receiving high-level requests from the scientist, and supplying high-quality prioritized data. From the perspective of Figure 3, this involves migrating parts of all ground operations processes onto the spacecraft to assure that it can satisfy such high-level requests while avoiding risks to the system's safety[5]. Issues such as security (instrument accessible only to authorized personnel for each context), could be handled by the middleware. Both the first and the second approach (improved downlink data quality) suggest the need for intelligent remote data handling, including techniques for remote science analysis and data-mining, data compression and fusion, and event-driven data handling. The third approach (opportunistic science) could be enhanced by improved leveraging of remote resources such as planning, scheduling, science processing, etc. More generally, these three approaches motivate accumulation of a set of reusable remote IT resources (such as intelligent data handling as mentioned), which can simultaneously address more than one set of needs. We call this "shared middleware", and address recent software trends that may make this feasible for exploration.

State of the Art

Historically, software applications were built directly on top of the platform's operating system. A later trend towards separating "user interface" from "business logic" and "data access" fostered n-tier client/server computing, and improved modularization of code. Later, abstraction of the operating system interface and improved networking technology made building such distributed applications easier, hence the recent growth of "standards-based" middleware. Issues such as performance, service level, and software architecture are addressed for particular applications to ensure that a proposed solution meets multiple objectives (including affordability).

Taking a "shared middleware" approach toward defining and building software services can dramatically simplify application development, thus enabling those approaches to improving access to Mars defined above. Modern COTS middleware can elegantly address such issues. For example, CORBA provides a service infrastructure with pluggable components, freeing applications from having to implement such features separately. Also, pluggable components are "replaceable" with alternatives performing the same function but providing additional features. For example, "remote method invocation" can be replaced by "reliable remote method invocation" with little or no need for change to the client application, as long as the request is properly satisfied (i.e., unchanged interface). Providing such "enhanced" capabilities is much simpler with standards-based common services and the component software approach.

An example service that can significantly simplify application development is the "CORBA Event Service". Using such a service lets the developer avoid having to implement his own event loop (for every application), in favor of "publishing" or "subscribing" to particular types of events in a standard fashion whatever the meaning of the "event". At the science application level, such a service could help make a remote planning decision based on subscribing to information about remote resource location and availability without operator intervention. This could dramatically reduce the 6-step cycle time described above and enable opportunistic science on the rover. Further, using *asynchronous* event services can make resource use even more efficient; for example, a requester can perform other useful work while waiting for the response, which is handled in a standard way (e.g., via a callback) when it arrives.

Middleware Applications at JPL

While this concept may appear far from reality for low-cost space missions, given today's confused middleware environment, with its apparent plethora of rapidly-obsolete standards, there are examples of this approach being successfully prototyped and used operationally in other contexts at JPL.

A first example application illustrates great simplification of client software by migration to a COTS-based service architecture[6]. Figure 4 shows a prototype re-implementation of the monitor and control information service (MCIS) deployed a few years ago in JPL's Deep Space Network (DSN) ground system. The original implementation was an entirely custom-built publish/subscribe service costing several work years to design, implement, test, and deploy. By replacing the underlying custom service with the CORBA event service (itself built upon other standard CORBA services), with a COTS implementation based on freely-available open-source in The ACE Orb (TAO)[7], most of this original code could be removed, without changing the client and server API. Apart from the benefit of significantly-reduced maintenance for the JPL-written custom code, and the leverage of 20+ work-years of effort in building TAO, other applications could then use standard CORBA event service (for very different purposes). Moreover, the service itself could even be evolved independently from the application code (e.g., to adapt to characteristics of an interplanetary vice standard terrestrial internet), again potentially benefiting all service users at once, and still with little or no change to application code or API.

Another JPL middleware-based application, called Shared Net, has been developed and is currently being field tested by the United States Marine Corps. It demonstrates how middleware-based services can be used to greatly enhance the capabilities of independent systems. The IMMACCS system[8] (of which Shared Net is a part) is comprised of a collection of several hundred sensors, users, intelligent agents, and existing data collection and analysis systems. These components are hosted on a number of different hardware platforms and are scattered across a wide geographic area. IMMACCS components generally operate autonomously, but occasionally need to collaborate with each other to share relevant information and to coordinate their activities. Likewise, the components occasionally need to download information to one or more

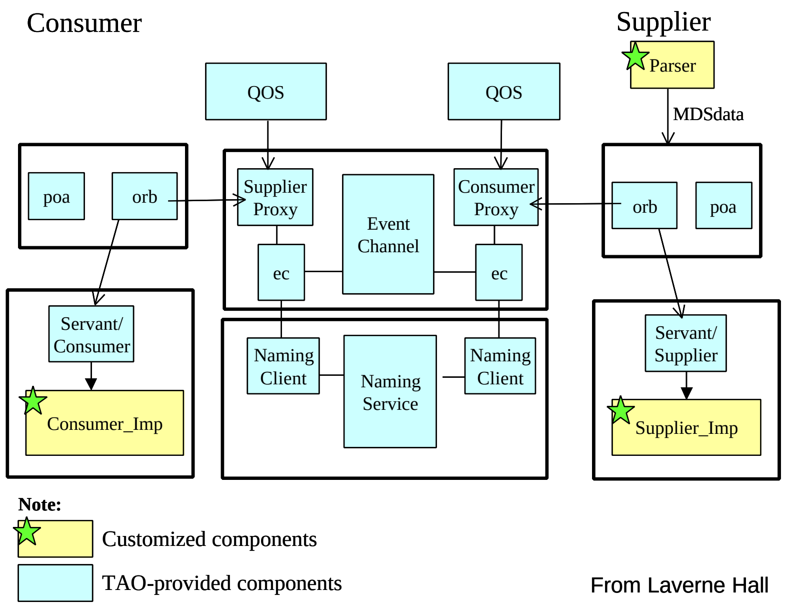

Figure 4 Example Monitor and Control Information Service Re-implementation

command and control centers, and retrieve new instructions (missions).

Like our situation with the Mars Reconnaissance Orbiter, IMMACCS components have far more data to share than the network has capacity. Since both IMMACCS components and the network nodes are constantly moving, the precise connectivity and capabilities of the network are constantly changing. For example, components located in an RF shadow (say a canyon) may only have connectivity when an airborne or low orbit relay is overhead. In some instances, a dynamic change in the configuration of the network requires the use of entirely different communications protocols.

Prior to Shared Net, the components were responsible for managing their own communications and users located in remote command centers generally mediated the communication between the components manually. The approach has a number of disadvantages:

- Each component must understand and adapt to the state and capabilities of the network. This is an exacting task, and conflicting approaches could be disastrous.
- Components are unaware of each other's communication needs and have no way of jointly arbitrating the use of the network. This can result in substantial delays in transmitting critical data while another is transmitting less critical data.
- Collaboration between components is limited by the long-haul connectivity to the command center. This may significantly delay the movement of critical information between components in close physical proximity.
- A translator is needed to translate between each distinct combination of data formats or protocols used by the various components.

Under the Shared Net approach, the various components share information via a set of common middleware-based data-management and distribution services. They can update the shared knowledge base by making changes to the common object repository and learn about information shared by others by querying the Object Repository, or by subscribing to the creation, deletion or modification of specific types of information (e.g., by subscribing to changes to the mission directives assigned to me). In effect, the publish-subscribe model provides a client (or user) with an information feed tailored to their specific needs. The implementation of the core information management services is intentionally transparent to the clients. This approach allows the services to adapt to a variety of different configurations and contingencies without requiring any changes to the clients. The core information management and distribution component is comprised of the middleware services shown in Table 1.

These services allow IMMACCS components to be plugged into the network much the same way in which hardware cards are plugged into a computer bus. Even though the components were written in different languages, by different organizations, and hosted on diverse CPUs and operating systems, the middleware approach allows the components to

Service	Description
Object Repository & Persistence	Maintains a rich object-oriented representation of the information and data provided by each component. For example, it contains objects representing physical assets with their location & capabilities, mission objectives, planned/actual traversal routes, weather readings, topology, and seismic sensor readings. Additionally, the service interacts with several different databases and file systems to save the information to non-volatile storage.
Publish & Subscribe	Dynamically manages the flow of information across the networks in order to satisfy both static system-wide priority policies and the quality of service required by dynamic subscriptions. It allows components to specify interest in specific types of information maintained by Shared Net (e.g., inform me when a mission objective has changed or when the weather data indicates an approaching storm). When a subscription is met, the Shared Net will pre-stage the information in the component's local cache. Subscriptions can also include a quality of service that indicates the relative importance of the subscription to the component.
Information Prioritization	Prioritizes information based on a set of rules. It coordinates with the communication services to determine which information should be transmitted given the available bandwidth.
Data Archival & Replay	Maintains a detailed history of events occurring on the server. This allows an interested user/system to retrieve and replay events that occurred during a specified period of time.
Aggregation	Uses a set of rules to summarize low-level data into a more compact form. For example, it may take a set of periodic sensor readings reporting the same data readings and create a new summary object indicating that the reading held over a period of time. This allows the summary object to be transmitted to interested users/systems while retaining the raw data.
Query	Allows users to retrieve objects that meet specified criteria.
Translation	Provides bi-directional translation between the component's native data format and the object representation. This service allows any system to send relevant information to any other system.
Fault Tolerance	Provides automatic & transparent replication of selected services and objects on either the same or different processors.

Table 1 Shared Net Middleware Services

efficiently share information and isolated them from knowledge of the current network configuration and status. In addition, they do not need to be aware of exactly who is providing/requesting the information nor of the specific information format or communication protocol used by the other components. This allows new components or services to be added to the system without requiring code changes to the existing components.

Not surprisingly, this architecture also provides an excellent environment for supporting agent-based expert systems. Agents are themselves objects (albeit with behaviors) that live within and collaborate with each other via the Shared Net. The common representation of information also allows the agents to "reason" about the information without needing to understand the particular data format used by the originating system (rather it uses the translation service to mediate).

4. MIDDLEWARE APPLICATIONS IN MARS PROGRAM

Both of the previously-described JPL middleware applications demonstrate the feasibility and benefits of developing middleware-based services to improve access to Mars as described in Section 2.

At a minimum, we can gain an unparalleled level of flexibility and adaptability by building a service-oriented architecture based on middleware. In some instances existing COTS middleware services (such as object repositories and persistence) will directly meet our needs. In other instances, particularly where our needs diverge from the standard COTS implementations, we can transparently modify the services to add features such as "reliable data transfer", "fault tolerance" or even adaptations to different network protocols. This approach also provides benefits such as improved reconfigurability for unforeseen future uses. Recently, the loss of a star tracker in the DS1 spacecraft did not result in catastrophic loss of navigation capability, because a camera could be reprogrammed to act as a star tracker. An on-board architecture capable of achieving this is already under development within JPL's Mission Data System[9]. We apply similar reasoning to the leveraging of distributed resources via middleware services.

However, as shown in the Shared Net example, middleware can also serve as an enabler allowing us to build collaborative (or at least cooperating) communities out of the independently developed systems that will be deployed on Mars over the next few years.

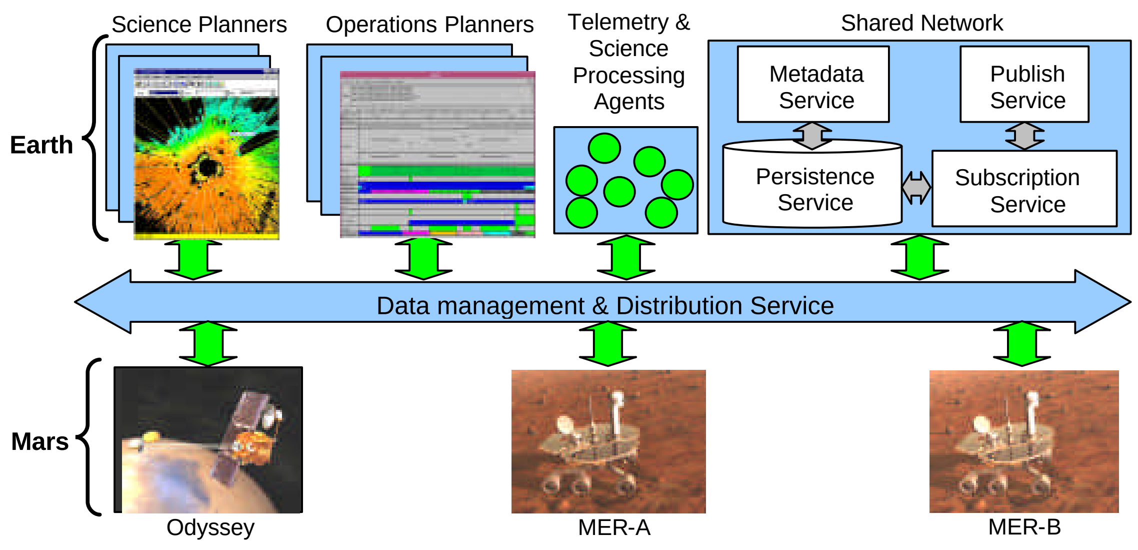

Figure 5. Multiple Mars mission software architecture based on the Shared Net enabled IMMACCS model. All components communicate via a common data management and distribution service, which relies heavily on middleware services

Interacting Missions on Mars in 2003

In 2001 Mars Odyssey starts orbiting Mars, collecting data with its three primary instruments, and beaming that data back to earth. Among these instruments, the thermal emission imaging system (THEMIS) generates the lion's share of the data[†]. This instrument can image the surface of Mars to a 20-meter resolution at 5 different visible spectral bands during the day, to a 100-meter resolution at 9 different infrared spectral bands during the day, and to a 100-meter resolution at 2 different infrared spectral bands at night. Since Mars has $1.45{\times}10^{14}$ square meters of surface, the three instrument mode data volumes are 8330Gb, 667Gb, and 133Gb respectively. Adding these volumes and supposing that Odyssey can downlink 70Mb per day results in discovering that it would take over 300 years to downlink all possible measurements. For this reason Odyssey will continue to collect THEMIS data even after two MER missions place rovers on Mars, and Odyssey picks up an extra data relay duty.

Within the 2003 time frame, Odyssey will make 3 different sensors available to the Mars Program: a gamma-ray spectrometer, a radiation environment sensor, and the THEMIS. At the same time each rover will make its suite of sensors available. While this suite has not been determined yet, research into similar "Athena class" rovers has involved 5 different sensors: a panoramic camera, an miniature thermal emission spectrometer, a Mössbauer Spectrometer, an alpha proton X-ray spectrometer, and a microscopic imager with a rock abrasion tool. While the data rates for these instruments are still unknown, the rovers are being designed to be able to traverse up to 100 meters per sol, and the Odyssey UHF radio can receive >100 Mbits over an 8 minute time window each sol when the orbiter flies over a rover. Thus Odyssey can receive over 200 Mbits per sol, but only pass on 70Mbits per day. While each rover can circumvent Odyssey and transmit directly to Earth, a rover's relatively small solar panel and antenna implies a much lower transmission speed. Thus the three missions will compete for Odyssey's downlink transmission bandwidth.

This interaction can be finessed away by strategically sequencing and all communications activities and amounts months in advance and scheduling other operations around the communications activities on a daily basis, but this approach causes a fair amount of inefficiency. Consider a scenario where MER-A is on a long traverse while MER-B has just found a rich source of data. Given a strategic communications plan for each rover to use 35 Mb of Odyssey's downlink bandwidth, neither MER-B nor Odyssey can take advantage of MER-A's wasted bandwidth. Even worse, MER-A has to stop its traverse and waste both its and Odyssey's time/energy to transmit even though it has little data to transmit. While this approach is better than having each mission run in isolation, it does not take full advantage of the multiple missions. Taking full advantage involves dynamically allocating

[†] Our data on Mars Odyssey comes from the "2001 Mars Odyssey Fact Sheet" available at the Odyssey gamma ray spectrometer team's website: http://grs8.lpl.arizona.edu/faq/

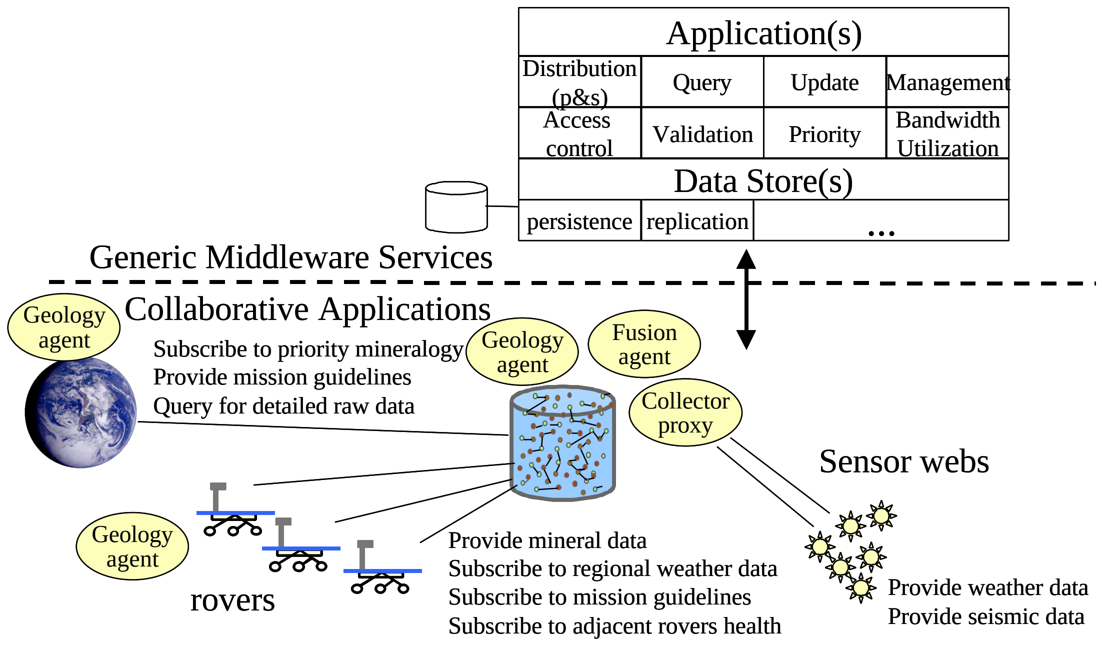

Figure 6. Example Software Architecture Enabled by Middleware Services

bandwidth to maximize the amount of downlinked data. This involves dynamically coordinating the missions' sequences as they evolve.

Middleware-Based Application Architecture

A straightforward middleware-based service approach (shown in Figure 5) could also be used to coordinate the applications. A set of generic services similar to those provided by Shared Net could be provided, but specifically tailored to our needs. For example, the software bus would need to allow components to transparently be plugged in either on Earth or in situ. Likewise, the Object Repository, and a suitably modified data distribution service could maintain and distribute the telemetry and commands for these assets both locally on Mars, and over the long-haul to Earth. Since the middleware understands the priority of the information and the current capabilities of the network, it effectively provides an intelligent store-and-forward capability mediating between Earth and the fielded resources. Priority information could be dispatched (and even preemptively retransmitted based on past transmission receipt behavior), with lower priority information being sent if bandwidth is available. Summarization agents could be used to collapse near identical data sets while allowing the scientist to later request the raw data download stored by the Data Archive service.

Supporting Autonomous Operations

Lastly, Figure 6 shows an example of a collaboration architecture that *builds* upon the same middleware-service approach. However, instead of merely collecting and distributing information, the middleware becomes a hub for collaboration where each resource (sensor web, rover, etc) utilizes services provided either by other resources, or by the infrastructure (data management, communication).

For example, a weather sensor web could regularly send out particulate readings (measuring the amount of material suspended in the atmosphere), which would be stored by the Object Repository. Individual rovers could subscribe to be notified if the particulate count in their adjacent geographic area exceeds a safe level. Upon receiving a notification, the rover could shield its optics. Note that the same notification could be triggered if an orbiter's camera (or other instruments) detected an emerging sand storm. Likewise, an atmospheric scientist might ask for individual readings to be transmitted at a low priority, but ask for summary information to be transmitted hourly, or more often if the deviations in the observations exceed a particular threshold.

Combined with *in-situ* planners, the collaboration approach could enable opportunistic science. For example, a rover's report on a particularly unusual (and unexpected) rock could result in the planner retasking the rover to refine the investigation, or even scheduling other resources (such as orbital platforms or rovers equipped with additional sensors) to investigate.

The flexibility of the architecture also allows contemplation of an “evolvable” set of such services, progressively implemented by successive Mars missions, each contributing capability to the overall Mars environment as a secondary goal to their primary (sensor) science goals.

5. PROPOSED PROTOTYPING APPROACH

In order to develop and validate the concepts described here, we have proposed a prototyping approach to progressively increase confidence in their readiness and applicability for near-term missions. We begin by sketching an initial simple testbed configuration, consisting of workstation and target platforms, the latter represented by a “flight” CPU (e.g., 200MHz Power PC), and perhaps a “science” CPU (e.g., a 400MHz Power PC or Pentium) in a VME card cage. This cage provides a “simulator” capability for testing and demonstration of simple applications built upon a simple middleware infrastructure (and could also use MDS components if available). At first, a basic communications layer would be provided by standard middleware (e.g., ACE/TAO), in order to demonstrate the feasibility of hosting such basic services on simulated flight hardware. Next, An initial proposed example application is WINDS, whereby a “local” model of the Jupiter atmosphere is built from successive measurements. The application would take simulated imagery from Voyager, Galileo or Cassini data, and image-processing algorithms (currently implemented on a ground workstation), and determining the feasibility of migrating such software to the simulated flight environment as a set of software components plugged into the middleware backbone. Generic application services (e.g., data management, publish/subscribe) would then be progressively added to this environment to determine the benefits of application flexibility and adaptability. The goal at this stage would be to measure whether additional applications become successively easier to implement into the evolving service environment, compared to the effort of building the first (standalone) application. If appropriate services are properly chosen for implementation, the effort to implement new capabilities should significantly reduce, as was demonstrated in the two unrelated JPL applications above.

As engineering hardware becomes available (e.g., FIDO rover), a more complex application would be implemented, e.g., MISUS as described above. This would be the first attempt to implement goal-based science objectives, and requires several services to be available in possibly-changing configurations. At this stage, some of the required components could remain on workstation platforms, interacting with the simulated flight hardware using middleware communications services – indeed, this would indicate the feasibility and problems of such a distributed approach without having to port each application component to the flight platform. The prototype testbed would then be ready to demonstrate middleware-based central planning, central science module, continuous planning and simulated on-board science analysis and platform control with a resource profile. Such ambitious application integration has been contemplated but not demonstrated for flight environments; the proposed prototype would therefore address many of the questions and concerns that have prevented this in the past, as well as determining the practicality of the middleware-service approach for such potentially-limited flight platforms (low CPU speed, small RAM, unreliable network). It also determines the likely timeframe when such applications would be feasible (based on simple prediction of the growth in flight computer resources over time).

6. CONCLUSIONS

We believe that middleware is heading towards commodity COTS product availability. Many market segments are driving this trend, and large enterprises expect COTS middleware to solve many of their large-scale application integration problems. The consumer market demands such supplier integration; requiring coherent information access and modification hardly conceivable a decade ago. We believe that such levels of integrated “science service” are attainable for space exploration, and indeed are required in order to make such exploration more affordable (i.e., reduced failures, increased return on investment). Coupled with higher-performance communication technology (e.g., optical communication) and processor speed, we expect feasible operation of an evolvable set of middleware services on the next generation of low-cost missions. Looking further ahead, we believe the esoteric technologies of virtual multimedia and even quantum computing may be harnessed to improve the effectiveness of our interaction with remote space environments, and to progressively engage the public in such science exploration.

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. The Shared Net development is sponsored by the Office of Naval

Research, and the Marine Corps Warfighting Laboratory. The authors express appreciation for the contributions made by the MarsNetIT study team at JPL in Summer 2000, supported by JPL's Center for Space Mission Information and Software Systems (CSMISS), and also for the support of the Mars Network Project Office.

REFERENCES

[1] R. J. Cesarone, R. C. Hastrup, D. J. Bell, D. T. Lyons and K. G. Nelson, "Architectural Design for a Mars Communications & Navigation Orbital Infrastructure," presented at the AAS/AIAA Astrodynamics Specialist Conference, Girwood, Alaska, August 16-19, 1999.

[2] A. Mishkin, J. Morrison, T. Nguyen, H. Stone, B. Cooper, B. Wilcox, "Experiences with Operations and Autonomy of the Mars Pathfinder Microrover," proceedings of the 1998 IEEE Aerospace Conference, March 21-28 1998, Snowmass at Aspen, Colorado.

[3] R. Sherwood, A. Mishkin, T. Estlin, S. Chien, S. Maxwell, B. Englehardt, B. Cooper, G. Rabideau, "An Automated Rover Command Generation Prototype for the Mars 2003 Marie Curie Rover," SpaceOps 2000, Toulouse, France, June 2000.

[4] R. Manduchi, S. Dolinar, A. Matache, F. Pollara, "Onboard Science Processing and Buffer Management for Intelligent Deep Space Communications," proceedings of the 2000 IEEE Aerospace Conference, March 18-25 2000, Big Sky, Montana.

[5] T. Estlin, T. Mann, A. Gray, G. Rabideau, R. Castano, S. Chien and E. Mjolsness, "An Integrated System for Multi-Rover Scientific Exploration," Sixteenth National Conference of Artificial Intelligence (AAAI-99), July 1999, Orland, FL.

[6] L. Hall, C. Hung, C. Hwang, A. Oyake, J. Yin, "COTS-based OO-Component Approach for Software Inter-operability and Reuse (Software Systems Engineering Methodology)" 0-7803-6599-2/01, Proceedings of the 2001 IEEE Aerospace Conference, Mar 2001 at Big Sky, Montana.

[7] D. Schmidt, University of California, Irvine (http://www.cs.wustl.edu/~schmidt/)

[8] J. Phol, M. Porczak, T. McVittie, R. Leighton, "IMMACCS - A Multi-Agent Decision-Support System," Technical Report CADRU-12-99, Cal Poly San Luis Obispo.

[9] D. Dvorak, R. Rasmussen, G. Reeves, A. Sacks, "SOFTWARE Architecture Themes in JPL'S Mission Data System", AIAA 99-4453.

BIOGRAPHIES

Anthony Barrett is a member of the Artificial Intelligence Group, where his research and development activities involve planning and scheduling applied to controlling clusters of spacecraft and managing the operations interactions between collaborating flight projects. He holds a B.S in Physics, Computer Science, and Applied Mathematics from James Madison University, and both an M.S. and Ph.D. in Computer Science from the University of Washington. His research interests are in the areas of planning, scheduling, and multi-agent systems.

Thom McVittie is a member of the Reliable Distributed Systems Group, where he focuses on development of software architectures supporting information collaboration across unreliable networks. Dr. McVittie holds a M.S in Computer Architecture, and a Ph.D. degree in Electrical and Computer Engineering from the University of California, Santa Barbara. His research interests include software fault tolerance, distributed systems, and agent architectures.

Norman Lamarra's current primary focus is on integrating Multidisciplinary Analysis Technology for NASA's Intelligent Synthesis Environment Program, whose goal is to develop an integrated engineering environment for the conceptualization, development, and operation of future space systems. Dr. Lamarra obtained the Ph.D. degree from UCLA in System Science (Communications Systems) in 1982. He has also worked for over 25 years in analysis & simulation of radar systems and phased-array antenna systems, and in real-time multiprocessing simulators.

Larry Bergman is the Manager of the Engineering & Communications Infrastructure Section, the Project Engineer for the JPL Supercomputer Facility, and the Program Manager for the Hybrid Technology Multithreaded (HTMT) petaflops supercomputer project. He obtained the M.S. degree from Caltech, and the Ph.D. from Chalmers University, Gothenburg, Sweden, both in electrical engineering. His research interests are in terabit optical networks and supercomputer technologies. He holds several patents and has authored over 100 papers.