Space Mission Operations DBMS (SMOD)

David P. Roland[*]
Computer Sciences Corporation, Moffett Field, CA, 94035-1000

Software tools for applications such as activity sequence planning, resource estimation, simulation, command preparation and verification, and resulting data analysis support space mission planning and operations. Historically, these applications have been loosely coupled via flat files in various formats. These usually require transformations between operational steps creating validity, consistency and accountability issues, slowing down communication, making working independently with subsets of data difficult, and overall increasing the effort of managing the mission. Recently NASA Ames Research Center (ARC) and the Jet Propulsion Laboratory (JPL) have collaborated on *Ensemble*, an integrating framework for supporting the ongoing Mars Explorer Rovers (MER) and the upcoming Phoenix and Mars Science Laboratory missions using a common Graphical User Interface based upon the Eclipse Open Source Development Platform. Ensemble uses a central RDBMS for storing the various data objects used in a mission. However, it still embodies the "file oriented" architecture of the previous MER approach, a legacy based upon such plan files of one or more sols' activities (a sol is one Martian day). This paper presents an alternative architecture focused upon a database repository and the lifecycle of an activity—proposal, scheduling, expansion, simulation, uploading, and execution. The key difference being the mission is viewed as a continuous set of activities rather than a set of plan files. Activities may be grouped in various combinations as required for analysis, scheduling, uploading or data review over whatever span is necessary. These state transitions are tracked and controlled using well tested and widely used relational database management system (RDBMS) techniques and architectural elements. The key architectural elements are described with the resulting derived benefits highlighted. Typical user concerns and approaches to mitigating user reluctance to these changes are discussed.

I. Background

Space mission planning and operations involve proposing, evaluating, scheduling, and uploading action commands and collecting, processing and correlating the resulting data products. Whereas mission requirements vary widely, the support problems and resulting solutions are remarkably similar. Some of these variations and similarities are examined below:

A. Mission planning timing cycles

Many factors affect space mission operations: the mission science goals, the spacecraft's instrumentation, the predictability of its location and the amount of control available.

1. Long range planning

Some missions have long preparation times and fixed scientific agendas For example, Cassini Investigation Scientist/Science Planning Engineer Dr. Kevin R. Grazier of the Jet Propulsion Laboratory, in an abstract for a lunch time seminar given Feb 24, 2004, describes the Cassini Saturn/Titan mission as follows:

> During its four-year nominal mission, the Cassini spacecraft will make nearly a quarter of a million observations. Understandably, then, the entire science planning process is a monumental effort. The observation planning for Cassini's nominal mission ostensibly ended over a year ago, meaning the bulk of the spacecraft's prime mission observations are cast in stone or at least rapidly-hardening concrete. Numerous factors have, however, dictated changes in the nominal mission trajectory, shifting several observations out of their "window of opportunity."

[*] Senior Computer Scientist, NASA Ames Research, Moffett Field MS 269-4, CA 94035-1000.

This quote says that even apparently fixed trajectory missions will experience late changes requiring new planning. The ability to select sections of a mission as a group, i.e., a *plan*, is a common requirement for mission operations.

2. *Daily planning*

Other missions require repeated rapid planning cycles. The Mars Exploration Rovers (MER) missions in particular required plans to be developed every half Martian day (sol) for each spacecraft. Teams of scientists review the previous sol’s downloaded images and data to decide on the next sol’s activities. This information is then combined with the engineering team’s needs for managing energy and vehicle safety, and communication windows to produce the best possible plan. For the MER mission, an artificial intelligence augmented planner is employed to aid the Tactical Activity Planner fit the maximum science into the typically over subscribed schedule.

The plan is converted into spacecraft commands, verified via simulation or execution on a lab vehicle, and sent to the Martian orbiter for relay to the spacecraft upon its awakening. Figure 1 describes the Planning and Operations workflow for the MER missions[1].

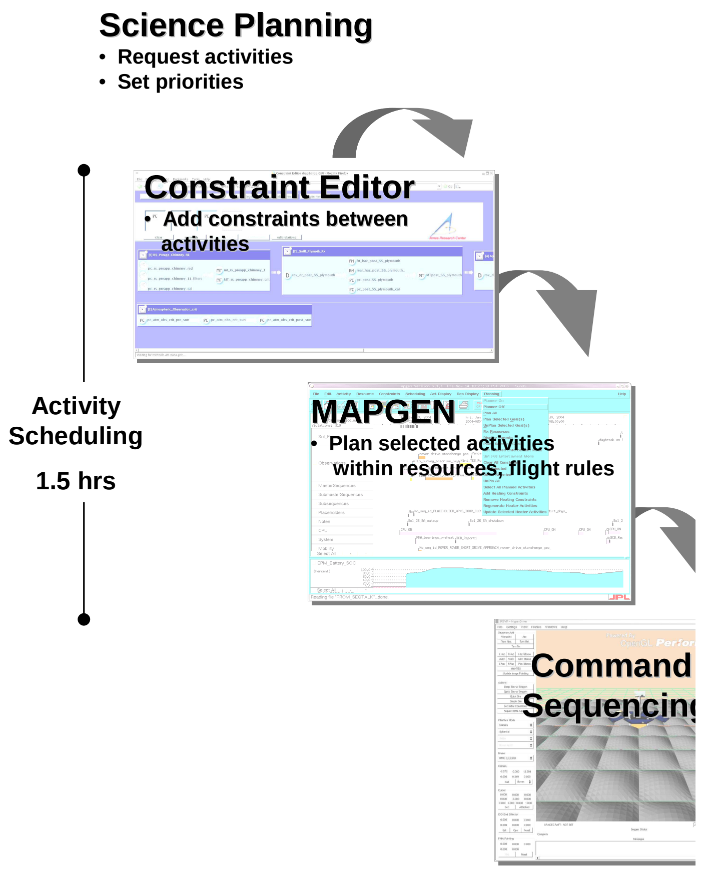

Figure 1 MER planning and operations upload process

3. *Common daily workflow*

Although these missions have widely varying planning horizons, the workflow is remarkably similar. The MER plans naturally fit into a daily framework, the Cassini mission also organizes its plans into daily segments. Figure 2 shows a daily timeline for the Cassini mission drawn from the complete mission plan.

This is reasonable considering most people’s work is measured in daily doses; “What will I do today?” It isn’t clear the data should be organized into these chunks? A MER file represents one plan, usually for one Mars day. The activities have unique identifiers but will appear in only one plan. A common practice is to create a new plan from pieces of previous sols and merge several sets of activities from various groups such as the science operations working group and engineering.

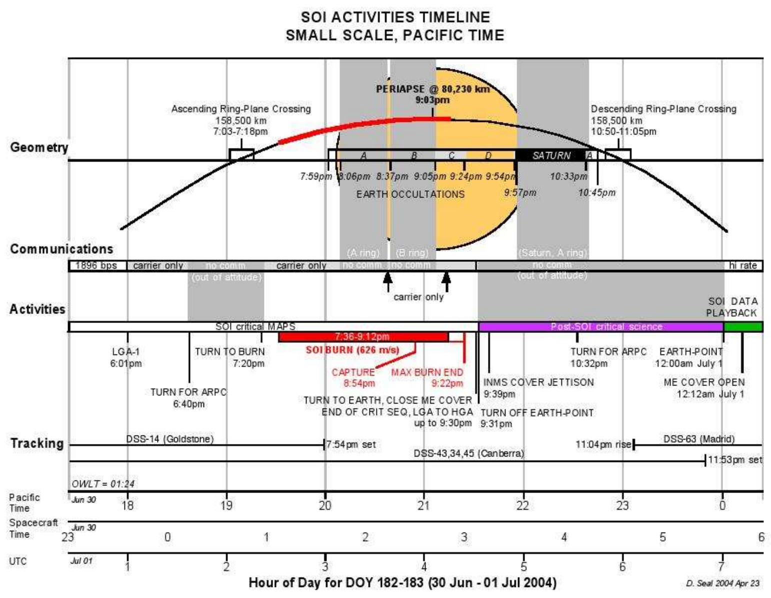

Figure 2 Single daily timeline for Cassini mission

B. Technology

Software technology has evolved over the years that space mission centers have developed planning and operations software. Each project’s desire to save cost and reuse trusted existing software has been tempered by the need to accommodate expanding spacecraft functionality that requires

ics

more capable support. New languages and technologies are often introduced only when new staff members decide the cost of modification exceeds the cost of replacement. Mission management is often suspicious but may be unable to counter the technology wave. One aspect that continues to influence the architecture of mission support software is the need to have many organizations involved.

1. Loosely coupled applications

Many software tools support mission planning and operations during phases such as activity sequence planning, resource estimation, simulation, spacecraft command preparation and verification, and returned data analysis. This software is typically written within each functional group, often using inputs from several sources, each with its own view of the data.

2. Files for data transfer

Historically, these applications have been loosely coupled via flat files in various formats. This inter-group communication often requires transformations between operational steps. The locally developed software written to do this creates issues of validity, consistency and accountability. The use of these files slows down operations, introduces opportunities for errors, makes isolating subsets difficult, and increases the effort to manage the mission. The Activity Planning Support System (APSS) for the MER's Spirit and Opportunity shown in Figure 3 is a good example of such loosely integrated mission support tools. Note the lines labeled APF, RML, SSF, CPF, etc.; each represents a different type of communication file.

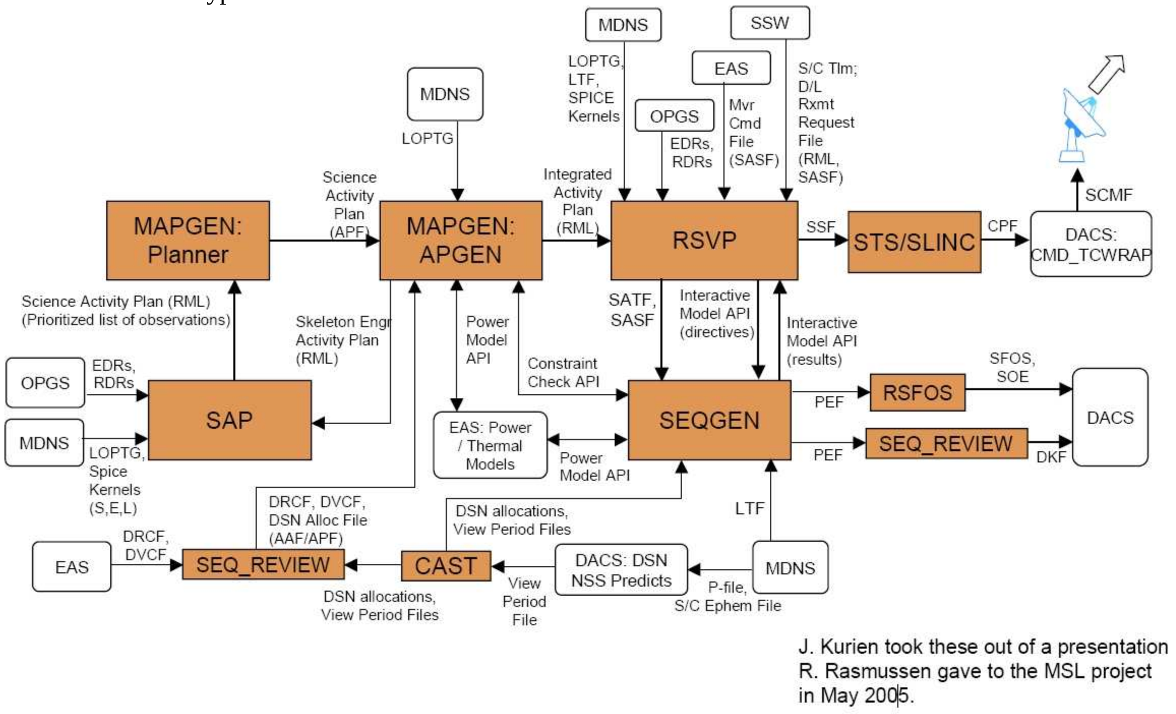
MDNS
SSW
EAS
LOPTG, LTF, SPICE Kernels
OPGS
S/C Tlm; D/L Rxmt Request File (RML, SASF)
MDNS
Mvr Cmd File (SASF)
EDRs, RDRs
LOPTG
Science Activity Plan (APF)
Integrated Activity Plan (RML)
SCMF
MAPGEN: Planner
MAPGEN: APGEN
RSVP
SSF
STS/SLINC
CPF
DACS: CMD_TCWRAP
Science Activity Plan (RML) (Prioritized list of observations)
Skeleton Engr Activity Plan (RML)
Power Model API
Constraint Check API
SATF, SASF
Interactive Model API (directives)
Interactive Model API (results)
OPGS
EDRs, RDRs
SAP
SFOS, SOE
RSFOS
PEF
MDNS
LOPTG, Spice Kernels (S,E,L)
EAS: Power / Thermal Models
SEQGEN
DACS
PEF
SEQ_REVIEW
DKF
Power Model API
DRCF, DVCF, DSN Alloc File (AAF/APF)
DSN allocations, View Period Files
LTF
EAS
DRCF, DVCF
SEQ_REVIEW
CAST
DACS: DSN NSS Predicts
MDNS
DSN allocations, View Period Files
View Period File
P-file, S/C Ephem File
J. Kurien took these out of a presentation R. Rasmussen gave to the MSL project in May 2005.

Figure 3 Applications workflow for MER missions

The literature on space mission planning and operations makes little reference to the details of this data transmission. An effort was made to use a standard Extensible Markup Language (XML) schema for defining the data results[2]. It's common for the communicating groups to simply have one deliver "what it has" to another that then converts it as needed. On MER, an XML based schema called Rover Markup Language is used in several phases but must be converted to another format for use in other tools. In addition, different phases use different subsets of RML and don't necessarily pass unused information through their phase.

3. IT analogy

The use of flat files and a mix of technology levels in various organization teams have an analogy in the information technology (IT) departments of most companies around 1970. At that time, the mainly COBOL shops using cards and multiple tape processing started converting to hierarchical fourth generation databases and languages. Within 10 years they moved on to online application processing (OLAP), relational database management systems (RDBMS) and the SQL language. This trend continues today with web browser based .NET™ and Java™ Enterprise (J2EE) applications. Although this technology shift seems to have occurred overnight, in fact it has taken the better part of three decades and was greatly accelerated by the need to fix the “Year 2000 problem”. No one should be surprised, then, that teams whose primary interest is controlling scientific instrumentation or analyzing downloaded results would be reluctant to rush into a software development using the latest “fad” approach, regardless its real or extolled benefits.

C. Recent developments

The MER mission software was mentioned earlier as an example of many tools integrated via file transfers. As successful as this approach has been, managers of the upcoming Mars Science Laboratory rover mission recognized that the interaction of these tools must be improved to support its greatly expanded capabilities. They have reviewed the MER process and prototyped new approaches that incorporate multiple functions within a consistent user interface using a central database. From a multi-mission perspective, this development, called Ensemble, has progressed to operational status on the ongoing MERs for some phases. Ensemble has made the following improvements over the original MER software:

1. Maestro/Ensemble Eclipse/Hibernate integration

The Jet Propulsion Laboratory authors of the Science Activity Planning (SAP) of the original MER ground support had begun extending it for MSL. They had moved it onto the Eclipse[3] plug-in development environment, renaming it Maestro[6]. It provides data review and targeting functions the scientists use to propose the next plan’s activities.

2. SPIF-e prototype – improved HCI; integration

NASA Ames’ Human Computer Interface Group analyzed the user interface of the previous components and MER work flow. The report suggested new graphical user interfaces and workflow that were prototyped as a browser application called SPIF-e. User tests with the prototype verified the improvements and suggested further development. The integration of the applications into a consistent framework was identified as a key improvement.

The SPIF-e project moved its development onto Eclipse and the resulting integration formed the foundation for Ensemble[7], an Eclipse-based Java integrated framework.

A key element of Ensemble is its RDBMS execution time database accessed via the Hibernate Object Relation Mapping system[8]. This database stores the activity dictionary read from ADML files and plans read from RML files. A plan can have new activities added or existing activities may be modified via the user interface and output as RML. However, external tools as the Europa mixed initiative planner/advisor and the APcore resource estimator have been integrated via inter-process communication without the use of files.

3. XML-based files – improve file parsing

The RML and its associated activity defining format, Activity Dictionary Markup Language (ADML), have been refined and made consistent. Although use on MER requires compatibility with the formats that were previously used, the upcoming MSL will allow for a consistent schema to be developed.

D. Data management architecture analysis

The evolution of mission planning and ground support has progressed from completely independent tasks reconciled at meetings to loosely coupled applications communicating via data files of various formats to integrated applications communicating via a central database.

Ensemble has mimicked the previous workflow and external file formats to enable it to subsume the existing software on the extended MER missions. RML files are read in and manipulated on the database and then written out as revised RML files. These RML files remain the repository and defining data for the daily plans. This approach makes introducing the new software easier as it requires little or no modification to non-integrated applications. It does raise issues of access or ownership for independent groups, however.

This is, therefore, a partial solution. Modern database technology has many benefits to offer for use on new missions. The role of the database can be expanded from a “run-time” or “execution” database used to hold the data during modification into that of the central data repository. The Chandra Data Archive[4] and the Columbia[5] mission seem to have taken this approach and made a central database the keystone of their architectures.

II. A Space Mission Operations DBMS (SMOD)

A. Multiple views of a mission

1. Plan or activity-centric repository

The Ensemble system retains the data structure that organizes plans as daily sets of activities and their associated data such as constraints that was used in the MER mission workflow. In fact, the *Plan* RML file contains sets of *Observations* that contain sets of *Activities*. Other sections contain constraint, sequencing and resource information, each provided by different groups. One person, the *Keeper of the Plan (KOP),* is responsible maintaining the integrity of this file that will ultimately contain the uploaded activity plan. Its filename usually identifies it with the sol and status, e.g., sol_052_activity_plan_final.rml for the result of combining the sol_050_sowg_science_plan-merged.rml and sol_052_activity_plan_final.apf. Reports such as sol_052_activity_start_and_end_times.txt are derived from the final combination.

This plan-centric organization has limitations. As the MER mission has far exceeded its original life of 90 days, longer duration plans have been developed. If a daily plan for sol X has activities that extend into the next sol, which file should it be stored in? If it's both sol x and sol X+1, how do you maintain consistency when it's modified in one or the other or prevent it from being sent to the spacecraft multiple times?

Multi-sol planning is also a requirement for the upcoming MSL and Phoenix Mars missions. Orbital missions, of course, have timing cycles related to their orbits and their orbited bodies' cycles. However, orbital missions often use the concept of a daily plan as well. Those missions will have fewer issues of duplication but the boundaries cause issues of splitting activities that extend through them.

2. Mission as continuum of activities

Rather then organizing the mission as a set of plans containing activities, a mission can be viewed as a continuous set of activities. Each activity has a life cycle of state changes as they are proposed, selected, scheduled, executed and analyzed, or not. This approach focuses attention upon the activities rather than their grouping into plans. Plans become a collection of activities that may span arbitrary durations and contain activities that are selected based on various criteria.

This concept is not unique, the Chandra Data Archive's architecture is similar.

3. Collections

Activities are relatively small units of work. Large tasks are combinations of activities in specific order, often with timing constraints between the constituents. On MER, these spanning collections are called *observations.* In most cases, the observation comprises activities that are to be treated as a unit where all or none are scheduled and executed., although that isn't an absolute requirement.

Other missions have discontinuous collections, for example, an atmospheric reading at regular intervals. On the Phoenix mission these are called *campaigns.*

Activities also have derivations. An activity may expand into a set of smaller activities. This is useful for planning at different levels of details. Sequences, the spacecraft commands, are the final product of the planning process.

One advantage of an RDBMS is its ability to create multiple views of data. A view can be a query or a joining of related tables of data.

4. Activity lifecycle

Activities have a natural, albeit variable, event-driven lifecycle with the following phases:

Proposed
Eligible for scheduling
Scheduled
Sequenced
Uploaded
Executed

Of course, a proposed activity is not always accepted for the mission. If accepted, an activity might not be scheduled and sent to the spacecraft. Even if uploaded, an activity may not execute because of local conditions or a last minute change. Moreover, if run, an activity might fail to create the data expected as shown in Figure 4.

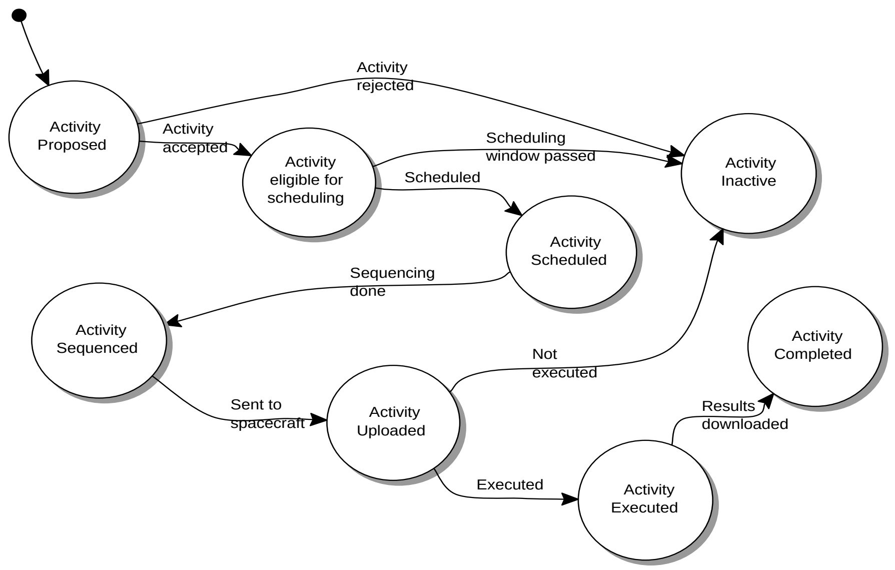

Figure 4 Simplified activity state diagram for MER missions

B. Object model

A flexible data management structure is

required to support the diverse data relationships mentioned above. Figure 5 shows an object model of such a structure.

Each node contains a set of sub-classed fields to contain its local data and a possibly empty set of child nodes. To implement a MER-like structure, the 0th (top) level node would be the plan, level 1 the observations, and level 2 the activities. To map the MER architecture, activities may decompose into activities or sequences. The nodal architecture allows for any number of children nodes easily supports this. Two level 3 children — sub-activity and sequence — are defined. Either or both may be empty or populated. The semantics of this arrangement would be "use the fields and sequences at level 2 for high level planning and the sub-activities' fields and sequences for detail planning".

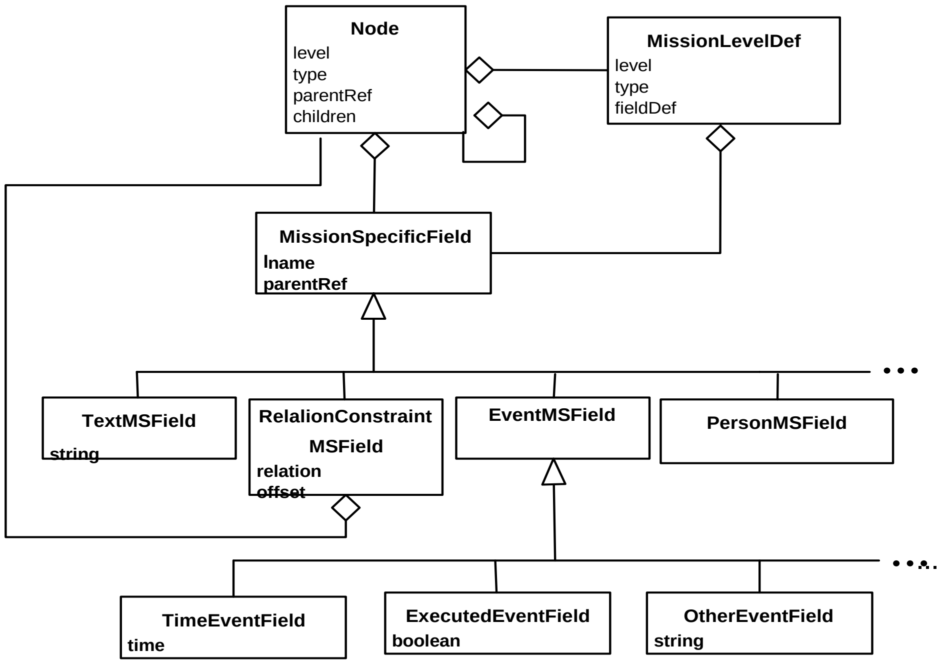

Figure 5 Simplified object model for flexible mission data structures

This structure is defined via auxiliary tables commonly called the *Activity Dictionary*. In addition to defining the levels, the dictionary would define the specific activity types and their associated data fields.

Implementing this structure in an RDBMS can be complex, requiring multiple joins to support the inheritance and collections defined. Using object relation mapping (ORM) software such as *Hibernate* relieves the developer from dealing with most of this complexity.

C. Benefits and aids to user acceptance

Introducing a database repository as the primary definition of the mission's planning and operations data elements provides the following key benefits:

1. Single definition of an activity

The overarching benefit is the single definition of each activity throughout its lifecycle. This activity may appear in many different views or be accessed and modified by many persons but it will always be the same, single representation of that activity. This greatly improves the integrity of the data and removes the need to merge files.

2. Multiple access

RDBMS repositories are the foundation of modern online application programming. Used with client-server Web-based interfaces or Eclipse-based Rich Client Platforms, RDBMS repositories have built-in facilities to control simultaneous access to data while preventing corruption. With the proper software controls in place, users would seem to own the activity exclusively even though others have simultaneous access. Many delays waiting for a file to be passed are removed. Specific features supporting multiple access include the following:

a. Change control and transaction audits

The data commonly handled with IT systems is usually sensitive and must have high integrity. This is accomplished with restrictions on who may modify data and recording when and by whom. In extreme cases, changes can be made conditionally and require a double signature.

The RDBMS uses a *transaction* to encapsulate the change operation. Only when all of the required data is collected and approved is the record committed to permanent storage and visible to others. Incomplete or interrupted modifications lead to a *rollback,* restoring the previous values. Both actions can result in an audit record.

Although text flat files may be maintained in a configuration management repository such as CVS, the entire file is updated each time any part is modified. Such systems can report the differences of each update; these tools are used only after the author decides to commit the changes, usually after many individual changes are made. If the file is not locked, others making changes will not be aware of the concurrent modifications until they update their changes, many times causing conflicts after merging that require manual intervention. Locking the file to the one user making changes serializes the access even though many users may want to make changes in separate, non conflicting sections of the file.

b. User authorization

The data can have associated access rules with users assigned various privilege levels. Using such privileges ensures that only authorized persons will modify the data. The application usually obtains the user‘s identity and offers only allowed functions. RDBMS features also prevent accidental or malicious modifications. This can make users nervous about getting access in critical situations but is mitigated with having a sufficient number of highly authorized staff available.

c. Tools integration

As noted above, ground data systems employ many tools in the planning and operations lifecycle. Integrating these tools, usually independently developed by different teams to accomplish a specific role in the mission, is a major benefit to the mission. Each development team wants to limit its efforts to using the minimum amount of input required from other groups. They'll rarely accept data that's just "passing through" to a later phase of the mission. Keeping the importer and exporters of such data current requires modifications to all of the applications in the chain.

While specifications such as XML help to organize the data and isolate applications from changes that do not interest them, the monolithic nature of such files make simultaneous access impossible. The RDBMS, while containing all of the various data elements, allows each application to extract only the components it requires. The effort to couple applications via an RDBMS is mitigated through the following techniques:

1) Application programming interfaces (API)

Programmatic access to RDBMS has become highly standardized through interfaces such as Open Database Connectivity (ODBC). The API approach provides the fastest and most complete data access. Many applications only require software modifications to their data reader and writer functions. These changes are minimized because of the following API features:

- Platform independence

The database server and the client (application) platform need not be the same. Since the introduction of the client-server architectures in the 1970's, access to the database has been through network connections to a program running continuously on a computer that need be known only through its name and interface port number. The advent of the web-browser has made remote databases available to authorized persons at almost any computer in the world.

- Language independence

The ODBC specification has been implemented as software libraries and drivers for almost every programming language and RDBMS vendor. An API such as the Java Database Connectivity (JDBC) combines with an ODBC driver from a vendor such as IBM, Oracle, Sybase, or the freely distributed postgres and MySql to enable Java applications to store and query data. Libraries in C and Perl provide similar interfaces. This language neutrality of the RDBMS is a powerful integration feature.

- Global access control

Mitigating the concern of wide availability is the access authorization available at many levels including username/password, hardware identification (dongle), or known client computer. Individual users may be identified with specific privileges or assigned to groups or roles with set access rules. Data can be controlled for modify or read only under access and program control. For example, authorized persons may modify an activity only until it has been uploaded to the spacecraft. After upload, the activity can accept input only from the telemetry system to record the receipt of results.

2) Importers/exporters to provide legacy interfaces

If the legacy applications cannot be modified, programs that read and import data to the RDBMS from legacy files and that access the data and write it to flat files of the required format can be developed. Although not the ideal solution, the repository remains the one true data source. If high integrity is needed, the RDBMS can "lock" the exported data until it is "returned" to the database, much like a software Configuration Management (CM) system would.

d. Variable planning horizons

A plan defined by query parameters for start and end times will have no restrictions on the duration. It will be the purpose of the plan that determines its length. For long term planning, a longer duration may be selected and passed into an automated planner. The resulting activity schedule may be selected in daily chunks to make reviews easier or for conversion to upload chunks that fit into communication windows. The integrated database allows for re-planning to begin at whatever time is required.

e. Flexible collections via queries

The RDBMS query language, SQL, and its corresponding ORM equivalents provide great flexibility for selecting collections of activities. In addition to using properties such as start date or instrument, associating tables can be defined that organize discontinuous chunks, i.e., campaigns, which can have any meaning to the scientists or ground system operators. New tools may have to be developed to make entering this information efficient. Some of this information, such as the following, is probably already defined for activities:

1) By state, sol-independent; the state of the activity (Proposed, Accepted, etc) would allow a query that returns all of the activities in a particular state, such as Accepted (but not scheduled).

2) By state, within a plan's duration; this query could represent a "What's in, what's out" report.

III. Conclusion

Modern Relational Database Management Systems allow for new ways of integrating the many software applications that support space mission operations. By redefining the organization of activities that define the mission and bringing together all of the stakeholders it is possible to gain the benefits of robust data, reduced preparation time, traceability, security and easy access.

Software teams may need training and support from RDBMS software experts. These people are common in the Information Technology community. Training in RDBMS integration is widely available to enable teams to participate directly. Ensemble, a NASA JPL and Ames Research Center collaboration is an example of using a RDBMS to bring together the many mission planning and execution functions in a common framework. This can be extended with the use of the RDBMS as the master repository of activities that move through a life-cycle rather mimicking the existing fixed duration plan file formats traditionally used.

The benefits of this architecture make the conversions and procedural changes more than worthwhile.

References

[1]Ai-Chang, M., et al, "MAPGEN Planner: Mixed-initiative activity planning for the Mars Exploration Rover mission", ICAPS03, Trento, Italy June 9-13, 2003

[2]Reich, L., "Binary Representation of XML Infoset in the Space Domain Position Paper", URL http://www.w3.org/2003/08/binary-interchange-workshop/14-ccds-w3cposition-updated.pdf [cited 14 March 2006]

[3]Eclipse, an open source community whose projects are focused on providing an extensible development platform and application frameworks for building software. URL http://www.eclipse.org [cited 14 March 2006]

[4]Rots, A. H., Winkelman, S. L., Paltani, S., and DeLuca, E. E., "Chandra Data Archive Operations", O*bservatory Operations to Optimize Scientific Return III,* Edited by P. J. Quinn, Proceedings of SPIE Vol 4844, p 172

[5]Zoeschinger, G., Wickler, M., Kohler, A., Axmann, R., "A Planning System for Payload and System Planning of the Columbus Module on ISS", SpaceOps 2004, Montreal, Canada

[6]Norris, J., Powell, M., "A Martian Eclipse", EclipseCon2005, Burlingame, California, 28 Feb – 3 Mar, 2005

[7]Ensemble Integrated Tools Demonstration, NASA Ames Research Center, Code TI, Contact James Kurien, URL http://ic.arc.nasa.gov/story.php?id=307&sec= [Cited 14 March 2006]

[8]Hibernate, a powerful, high performance object/relational persistence and query service. URL http://www.hibernate.org [cited 14 March 2006]