Moving Away From Ones and Zeros, Designing a Ground Data System based on Higher Levels of Abstraction

Michael Tankenson[1]
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, 91109

Previous JPL ground systems have been designed with the Ground Data System (GDS) engineer in mind. The focus on these systems has been on packaging and delivery of low level information (frames, packets, telemetry values) to the end user. It was not that long ago when project teams would be huddled over a workstation, examining crude displays of telemetry bits organized in various ways, trying to determine the status of a spacecraft. Understanding the data often required additional levels of GDS expertise, or worse, transformation of the raw data into alternative formats followed by ingestion into other tools so that the data became meaningful. The primary focus was often to answer these types of questions: "*Why did this particular frame fail Reed-Solomon decode? Why did this packet get marked as invalid? Why am I missing a block of telemetry from my query?*" -- which are completely valid questions to ask from a GDS Engineer's point of view, and large families of tools have been designed to help answer these questions. But these are not the questions that most users care about – which are more like: "*Why is the battery state of charge trending down? Show me a summary image report for the last traverse to the target. Show me a data accountability summary for the last DSN pass.*" Answers to these questions, which are what users are looking for, requires a higher level of abstraction and supporting tools than mining through ones and zeros. JPL has created a next generation capability called the Mission Data Processing and Control System (MPCS) which is designed to support this higher level of abstraction by providing customizable views of the ground system combining collections of lower level information into more meaningful ways. Instead of examining frames, packets, and individual telemetry data points -- MPCS is capable of providing comprehensive summary reports, product status, overall flight/ground event status, as well as payload health summaries. Based on these higher level views, end users can make tactical or strategic decisions, or drop into detailed analysis as needed. System designers need to continue building systems that support low level GDS troubleshooting – but the basic design of a GDS should be geared towards what end users actually need to see. This paper will describe the capabilities of MPCS that directly support these higher levels of abstraction, and which are being used today in missions such as the Mars Science Laboratory and other NASA missions

I. Introduction

Traditionally, a Ground Data System (GDS) is designed by people who understand GDS issues, based on requirements, design, and operational concepts that meet a missions needs. But the focus of the designers is normally on the low level details of the GDS (frame, packet, telemetry values, and commands). This harks back to a time when we did not really understand how to build such systems, and creating a Command/Telemetry system to fly missions into deep space was a challenge. We are beyond that challenge for the most part. There are literally dozens of GDS solutions (government, industry, COTS) that can be used to satisfy such needs. The challenge is no longer to figure out how to build the commands and sequences, or process telemetry – the real challenge is how to

[1] Development Manager, Multi-mission Ground Systems and Services (MGSS), MDAS Element, JPL Mail Stop 264-221, Michael.Tankenson@jpl.nasa.gov

design a GDS that presents information to users in a form that is immediately usable and meaningful. Moving away from ones and zeros means changing the focus of GDS designers, coming up with ways to create the products, reports, displays, and summary views that will provide information at the right semantic levels.

II. AMMOS Legacy System Behavior

JPL's multi-mission ground data system, the Advanced Multi-mission Operations System (AMMOS), has been successfully used for almost 20 years now, and has supported many dozens of missions. It was designed in the mid 1980s, its first mission was Magellan in 1989, and since that time has grown and expanded with additional capabilities and features. During this time, many valuable additions have been added to the AMMOS (Navigation, advanced Sequencing, CFDP, etc). AMMOS was a ground-breaking attempt to create a multi-mission system for command/telemetry support of deep space missions. Primarily created by GDS engineers (people with strong GDS backgrounds), there was little thought included for the non-GDS teams of people that would use AMMOS. Human Factors was basically non-existent at that time.

One of the results of that early AMMOS system, and a trend that continues to this day – is GDS behavior that is focused on low level command and telemetry processing details, together with valiant attempts to add additional, more "user friendly" features on top of them. Typically this results in a string of telemetry processing software, chained together with transformations of the data into a final form that is usable for the end user. The final end user tool might be something more "user friendly" such as EXCEL, Matlab or similar. It is not uncommon to require GDS experts in the loop, actively helping users to get at the data of interest.

III. The MER GDS, A First Attempt at Moving Away from Ones and Zeros

The Mars Exploration Rover (MER) project at JPL was one of the first missions to attempt to move beyond the traditional approach for telemetry analysis, and towards a higher level of abstraction. This was driven by the nature of the changes between Launch/Cruise/EDL and the Surface phases. During Launch and Cruise, MER was a traditional mission – lots of data in the form of packetized engineering, a few events (EVRs), and very few products. Operational support for this traditional type of mission was via realtime monitoring – AMMOS legacy processing and displaying telemetry items grouped by subsystem.

Grouped together, MER users would analyze information from displays, and depend on their collective experience to put the puzzle together. In the Telecom Subsystem example in the figure below, one can see raw telemetry changing, data represented in both Data Number (DN) and Engineering Units (EU) over a stick figure flow diagram. Other MER teams would be performing similar activities, sometimes using special tools to transform the data into a more usable form.

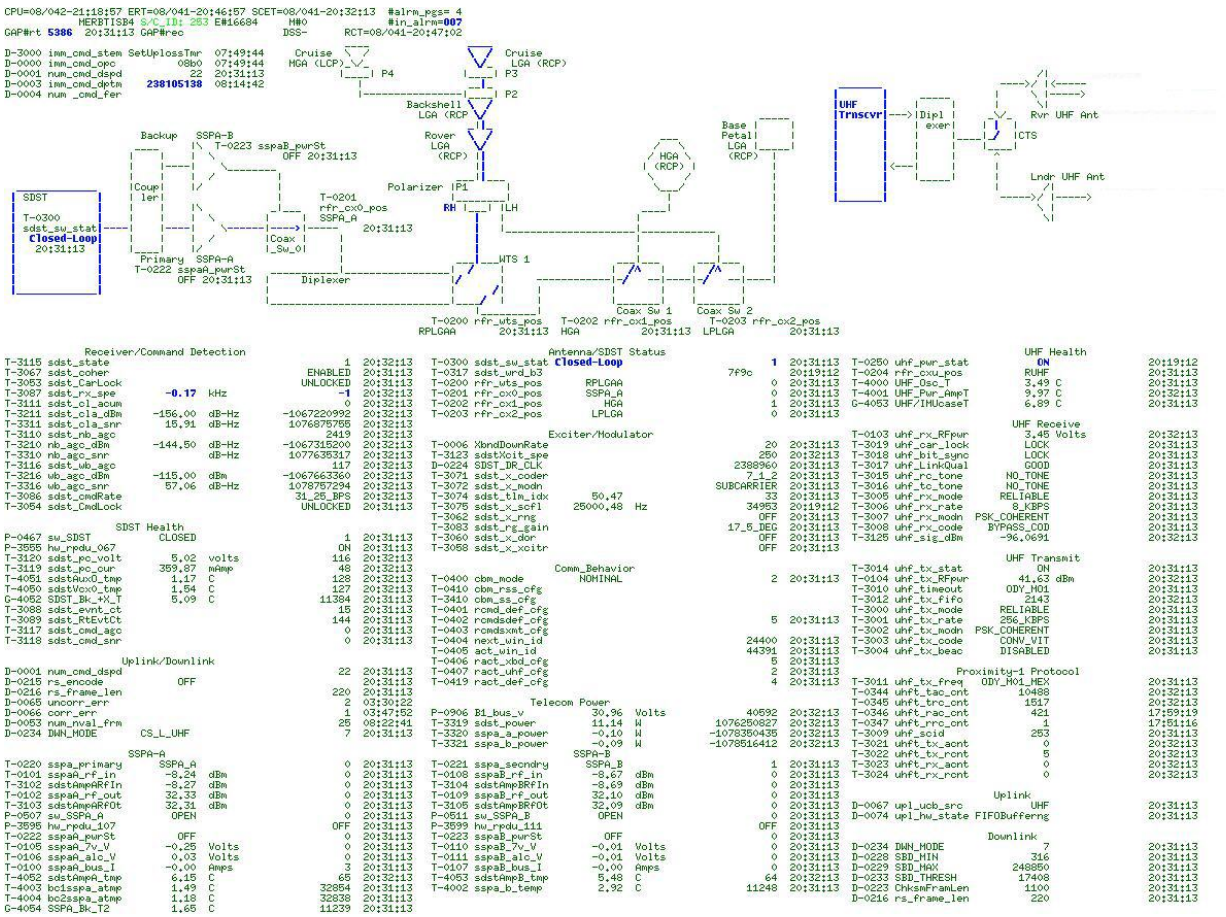

Figure 1. MER Legacy TLM Views

On the surface of Mars, things changed. MER tactical planning (which determines the activities to be performed the following day) required an army of engineers to produce the right products (TLM views, image products, uplink plans) in order to get ready for the Mars activity or SOL. By the time the MER extended mission phase started, it did not take long to realize that significant changes were required to produce the correct products, at the right level of abstraction, and to automate the turnaround of these products needed for tactical planning. The result was automated production of high level summary reports, which could be used in lieu of raw telemetry, and daily 'activity' reports, which combine key telemetry and thumbnail images into meaningful information. Using this approach, MER was able to quickly streamline operations and reduce the staffing considerably. The figure below shows the MER staffing changes between the Primary and Extended mission phases – much of which was possible due to the GDS changes.

	Primary	Extended	Savings
Overall Staffing	~300	~100	~66%
Health Assessment	24 FTE	1 FTE	96%
Data Management	12 FTE	½ FTE	96%
Planning Cycle	14-16 Hours	8-10 Hours	43-60%
Work Week	24¾x7	8x5	sustainability

Figure 2. MER Automation At Work

Another key data point from the MER example is how the nature of the data itself changed. Rather than streams of realtime engineering and EVR data, the Surface phase of the mission turned out to depend much more heavily on products and events. Realtime monitor of the telemetry displays essentially stopped altogether, since all MER data was delivered via a relay spacecraft, and so mission operations scenarios for downlink became a task of automating the processing, in non-realtime batch mode.

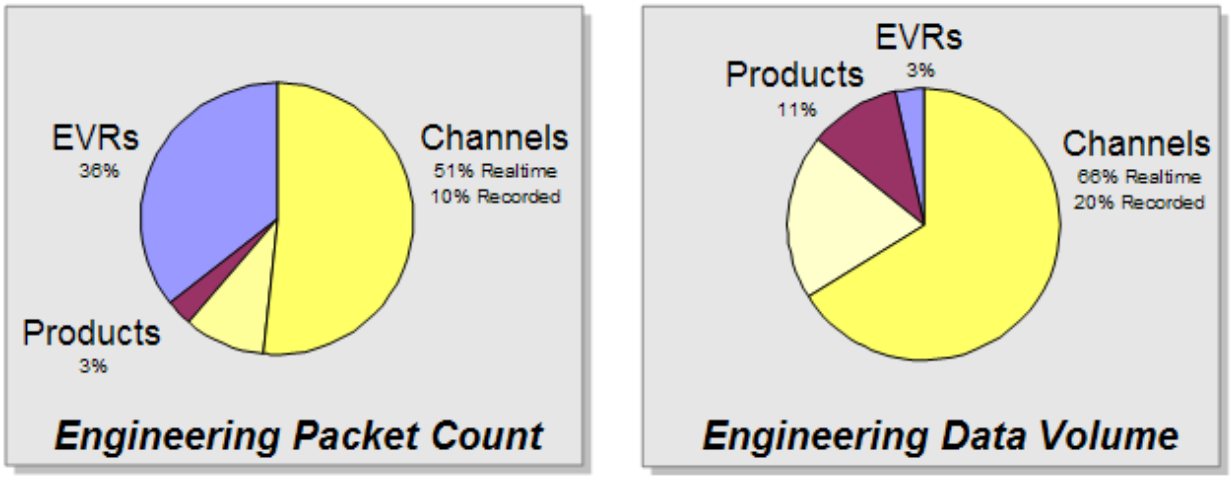

Figure 3. MER Data During Cruise Phase

During the Cruise Phase, mission operations was focused on realtime monitor of channelized engineering and events, with the majority of the data being channelized engineering. You can see the overwhelming percentage of engineering and event data in the Cruise Phase charts above.

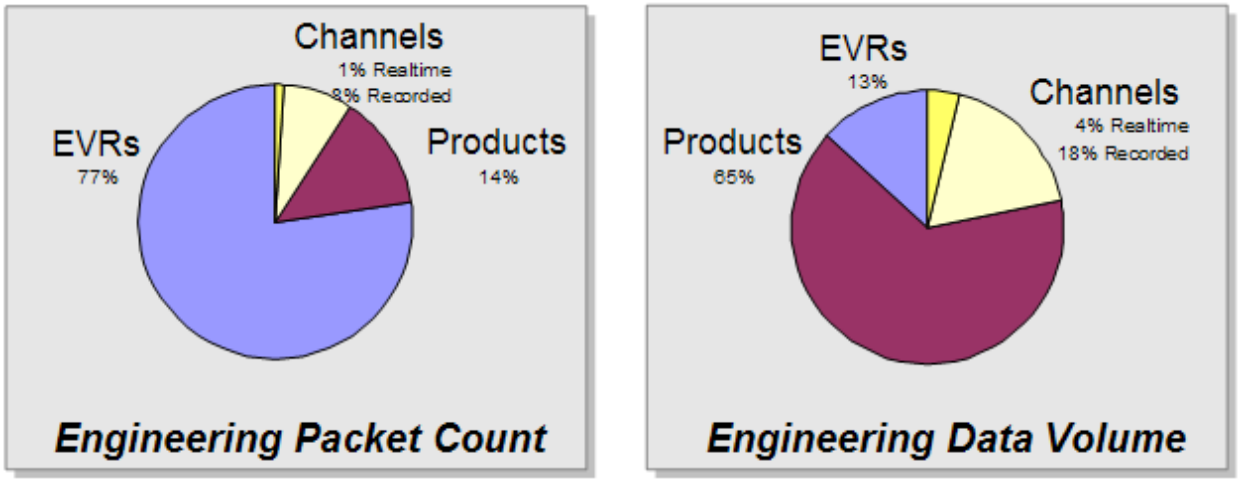

Figure 4. MER Data During Surface Phase

In the Surface Phase charts, the mission operations focus changed to a non-realtime mode, with most of the data consisting of products and events. MER's great success during the Surface Phase was due to many things, including an exceptional operations team. In addition, significant enhancements were made to the GDS to be able to produce high level summary reports – containing information at just the right level of detail to allow critical decisions to be made. The Rover Motion History Report example below, summarized a bunch of raw telemetry and provided a motion history product for the MER Mobility Team. The following example, a MER Downlink Summary Report, consisted of several collections of telemetry and thumbnail images, producing a high level product that was used to convey information to the MER Mission Director at the beginning of each SOL. Note that in both of these MER examples, the reports were custom-made.

Motion History Report

Request type: **NOTIFY_POS** *(run until all mechanisms reach commanded positions)*
Request status: **FAILED**
Request duration: **1.564 secs** *(limit = 18.156 secs)*
Motor controllers enabled: SCLK = 153885671.708

Motor	History	Start Position (radians)	End Position (radians)	Start Encoder	End Encoder	Start Potentiometer (radians)	End Potentiometer (radians)	Goal
MTES_EL	mothist-MTES_EL.spc	-3.04599	-3.33920	20953	22970	N/A	N/A	-3.65644 radians

Motor	Tenv (deg C)	Tcase (deg C)	Trotor (deg C)	Starting Climit (amps)	Running Climit (amps)
MTES_EL	-30.79 -30.79	-32.19 -32.19	-31.98 -31.83	0.581	0.515

Motor	Reason for stopping
MTES_EL	STALL *(motor stalled)*

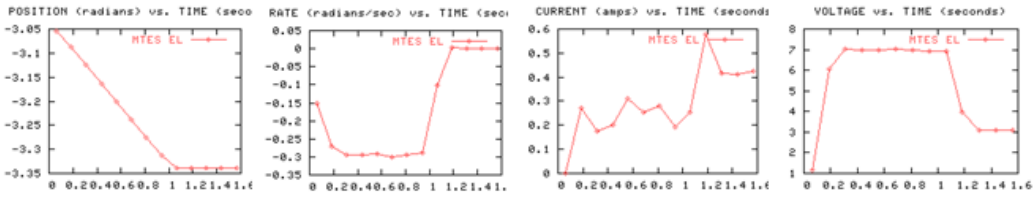

Figure 5. MER Example Motion History Report

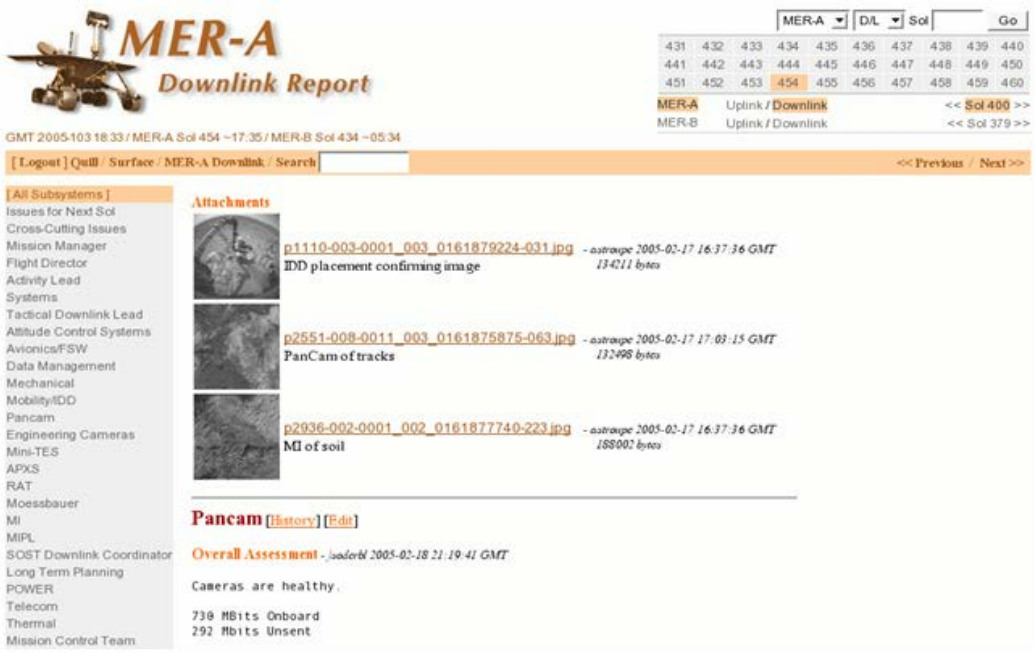

Figure 6. MER Example Downlink Activity Report

IV. Mission Data Processing and Control System (MPCS), Taking the next step

The Mission Data Processing and Control System (MPCS) is a new AMMOS product line focused on mission control and testbed functions, with strong support for early Flight Software Development (FSW), testbeds, and ATLO. MPCS provides multi-mission support for mission control functions used to monitor the health and status of JPL spacecraft, and is designed to be used during all phases of a project – from the earliest Flight Software (FSW) development phase, to high fidelity Testbed phases, to ATLO, and through Flight Operations phases. Providing such a system early in the flight software development phase means that the version of the ground system that FSW developers are using to exercise their flight software modules early is the same ground system that the mission will be flying with. This extends the tradition Fly-As-You-Test paradigm to new levels never before used, at least at JPL.

While the MER attempts to produce high level summary products turned out to be successful, the GDS solution at the time was crude, still requiring a number of manual steps, and was not truly multi-mission. MPCS components are Java-based, platform independent, and are designed to consume and produce XML-formatted data, with strong inheritance from previous flight projects (e.g. Mars PathFinder, Mars Exploration Rovers, Mars Reconnaissance Orbiter, DAWN). The MPCS component architecture is based on an event driven model with Java Messaging System (JMS) messaging used to trigger events and status. This JMS event driven architecture enables a plug and play architecture for additional components, especially those focused on telemetry, event, and product display. Some of the major architectural aspects of MPCS include:

- Platform independence (Linux, Solaris, MAC OS)
- Easy table-driven adaptation for new missions
- Support for "Event-Driven" Operations
- An end-user focus, including support for high level summary products (automated and on-demand)
- Adherence to NASA and industry standards (XML, SQL, JMS)

The platform independent nature of MPCS is what has allowed the MSL Flight Software Development team to use a high fidelity ground system (the same system the mission will fly with in late 2009) for early development.

The MSL team has selected Linux as their development platform, and deploy MPCS alongside the critical FSW environment. Obvious benefits include increased agility in development, quick turnaround of Command/Telemetry/Product/EVR dictionaries (the actual requirement on MPCS is 15 minutes or less), and continuous integration – allowing rapid checkout of the flight/ground interfaces.

MPCS software is essentially table driven which allows for easy adaptation for new missions. There still remains a small number of software components that require mission unique changes, but these are localized and easily supported. MPCS has been used for MSL, MER, and a small instrument on LRO called DIVINER.

The event driven nature of the MPCS design is a major improvement over the legacy system that supported MER, and the dozens of previous JPL missions. In MPCS parlance, an 'event' can be initiated from either the flight system or ground system. Flight events take the form of EVRs (Event Records), special packets that contain representations of critical flight system events

Flight Events

- Sequence started, Sequence completed
- FSW Event Record (Informational, Warning, Fatal)
- Key flight system activity started

Ground Events

- Start of DSN Pass, End of DSN Pass
- Product Complete, Partial Product Complete
- Channel in Alarm

These events take the form of JMS messages, which get published on the JMS bus. MPCS has been designed to monitor these event messages, and in turn can take action on them. For example, an end of DSN Pass event can trigger a number of actions – processing of products (complete, partial) for that pass, generation of summary reports containing information on Commands, EVRs, key telemetry. MPCS can generate event triggers for downstream processing by the JPL Imaging Processing system, allowing these steps to be automated, so that final calibrated products can be produced and used by the tactical and strategic planning teams. This support for flight and ground events, together with direct support for Report Generation and template-driven output from MPCS queries, allows users to produce the higher level of abstractions that they need to do their work.

MPCS is designed to simplify operations. Ground systems do not have to be overly complex, and users should not have to depend on GDS experts to get their job done. Unlike the legacy system it is replacing, MPCS tries to eliminate many of the mundane steps required to setup and run a GDS. For example, a single Operator Interface is used to configure and launch the MPCS core applications. All MPCS software is designed to use the same basic set of parameters, so that once a user has mastered a single tool, they have mastered them all. MCPS can be used in graphical mode or via command line parameters.

The figure below depicts a generic MPCS architecture, similar to the one being deployed for the MSL project. A brief summary of the MPCS component descriptions in the table that follows the figure. The basic MPCS architecture is based around a closely coupled set of downlink components using a high speed internal bus, with standard interfaces to the FSW target (development, testbed, ATLO, flight). All data is stored within an SQL database for later query, and selected information is funneled via a portal to the JMS message bus. Separate applications such as the telemetry monitor plug into the JMS bus and subscribe to the messages of interest. MPCS will deliver a basic set of telemetry monitor functions, but the architecture allows future applications to use the plug-n-play nature of the JMS bus to provide advanced display features.

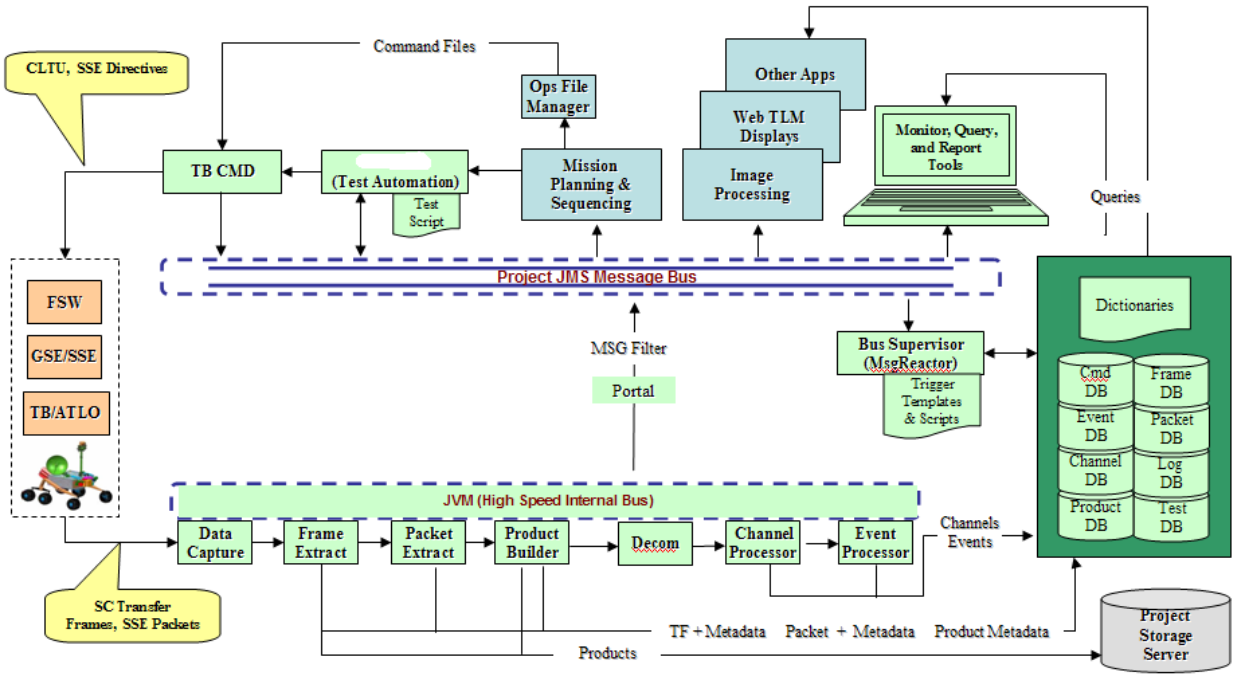

Figure 7. MPCS Generic Architecture

Table 1. MPCS Component Descriptions

Data Capture	Data Capture from different Flight System targets, depending on the mission phase. These targets can be individual FSW development workstations, called TestSets, or high fidelity Testbeds, or ATLO, or Operations (Cruise, Surface).
Frame and Packet Processing	Frame and Packet Processing for each of the targets.
Product Building	This provides a CFDP-compliant product builder for unacknowledged mode downlink.
Packet and Product Decommutation	Packet decom is the normal functions. Product decom is a new capability, able to selectively decom portions of a product as well as channelize selected values from a product
Channel Processing	Replaces the legacy channel processing functions (DN to EU, alarm limit checking, etc). This is fully decoupled from the Channel Display functions provided by Telemetry Monitor
Event Processing	This function supports processing of both SC and ground events. SC events are in the form of EVRs. Ground events are in the form of Test/Session actions (StartOfTest, EndOfTest, SyncStatus, etc)
Dictionary Management	This provides a comprehensive set of tools for managing project dictionaries (CMD, TLM, EVR, Product, etc).
Report Management	This provides standard templates and style sheets for creating reports, both automated and ad-hoc. Currently based on Velocity and JasperSoft templates and style sheets.
Life Of Mission Storage	This provides life-of-mission catalog service for Frames, Packets, Products, Channels, EVRs, commands, and associated dictionaries
Telemetry Monitor	This is the companion to Channel Processing, and provides visualization functions (list, plotting, alarm notification, etc)
Test Automation	This is a comprehensive Test Automation tool, which monitors message events and triggers actions based on those messages.
Testbed Interfaces	This provides the interface control for the DSN Emulator used on the testbeds, as well as interfaces to the GSE/SSE elements.

The current generation of MPCS displays are more evolutionary than revolutionary, and in some ways are required to mimic the AMMOS legacy behavior and look-n-feel (Lists, Plots, Fixed Pages). MPCS has enhanced these capabilities to support GRIDS and a flexible configuration of display preferences, including powerful filtering. In a typical MPCS data flow, realtime data is published on a JMS messaging bus by MPCS telemetry processing components, and then ingested by MPCS telemetry monitor tools which subscribe to the data of interest. In the figure below, an MPCS is showing four displays on a single tabbed window: Scrolling EHA messages, 2 List Pages, and a Plot Page – all of which are completely configurable by the user. The MPCS display design allows multiple display tabs, multiple collections of displays, with any combination of display types within a window. MPCS displays can be grouped within a single window with tabbed control, or the user may select separate windows.

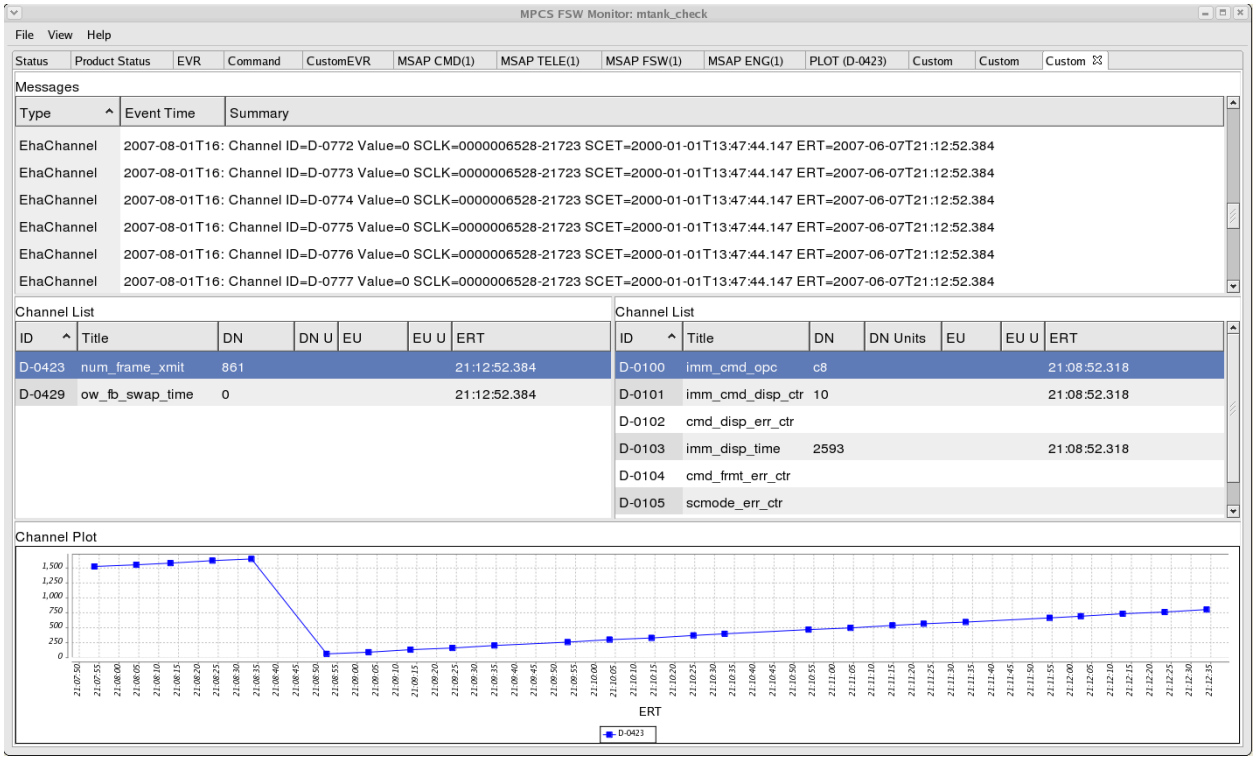

Figure 8. Typical MPCS Displays

The ability to produce high level summary reports is a key aspect of the MPCS design and the key focus of this paper. The MER examples shown previously are good representations of the ones that will be produced by the MSL teams using MPCS. MPCS supports a template approach, using Velocity Templates and Style Sheets for formatting data from queries, or JasperSoft libraries for creating advanced custom reports. Velocity and JasperSoft allow users to get access to any of the metadata from the MPCS SQL database, query the data of interest, and format the output as they please. JasperSoft allows a user to define a report template using a Jasper-specific template editor. It can then generate several types of output from that template, including PDF, XML, and HTML, and is suitable for creating web-browseable reports.

MPCS query tools have the same set of command line parameters and help/usage messages. All the tools support a common set of output formats:

- KeyValue (Keyword=Value)
- OneLineSummary
- Csv (Comma-Separated-Value)
- XML

- JasperSoft (Customized)

The examples below show several simple ways that users can transform their desired output, without having to resort to separate tools and/or GDS experts.

This command will get all of the EVR information from a particular test, and format the output as Comma-Separated-Value (CSV):

get_evrs –testKey 100 –o Csv

While this example will format the same data as Keyword Equal Value:

get_evrs –testKey 100 –o KeyValue

While this example will format a query on MPCS session data and format the output as a JasperSoft PDF file:

get_sessions –jasperReport KeyValue –outputFormat pdf –outputFile myreport.pdf

The output formatting is completely template-controlled (using Velocity or JasperSoft), which also allows users to create their own custom template and 'drop' it into the MPCS runtime environment. Reports are examples of the higher level of abstraction that MPCS is aiming for. The concept is that summary information about events (DSN pass, key EVR event, key ground event such as Product Complete, and so on) can be used to trigger the automatic creation of summary reports. Users will rely on this summary information to tell them what is happening on the spacecraft, and drill into the details if they have to.

Below is an example of a simple Downlink Summary report that is triggered by the End Of Session event. When the End Of Session event occurs, it triggers the creation of this report which summarizes the raw data collected and processes during that session. The current example is not terribly exciting (for example, the MER reports are much more comprehensive). What is important here is that these reports are generated automatically upon the receipt of triggers. Creating advanced reports using Velocity and JasperSoft will come soon.

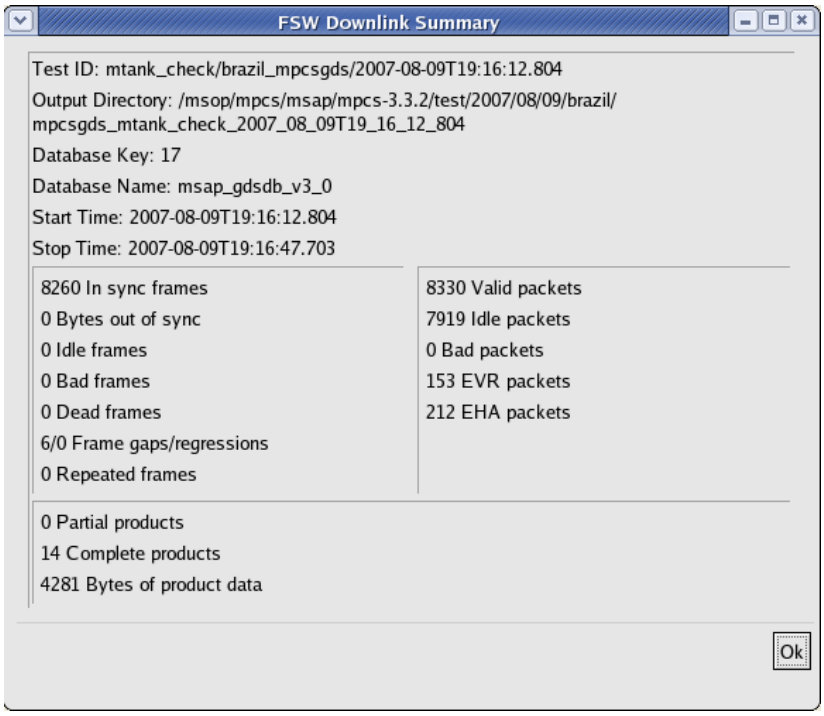

Figure 9. MPCS Example Report (Downlink Summary)

The End Of Session report is another simple example, with a summary of Products, EVRs, and selected telemetry channel information for the user. All MPCS reports can be generated automatically based on trigger event, or on-demand as the user needs them.

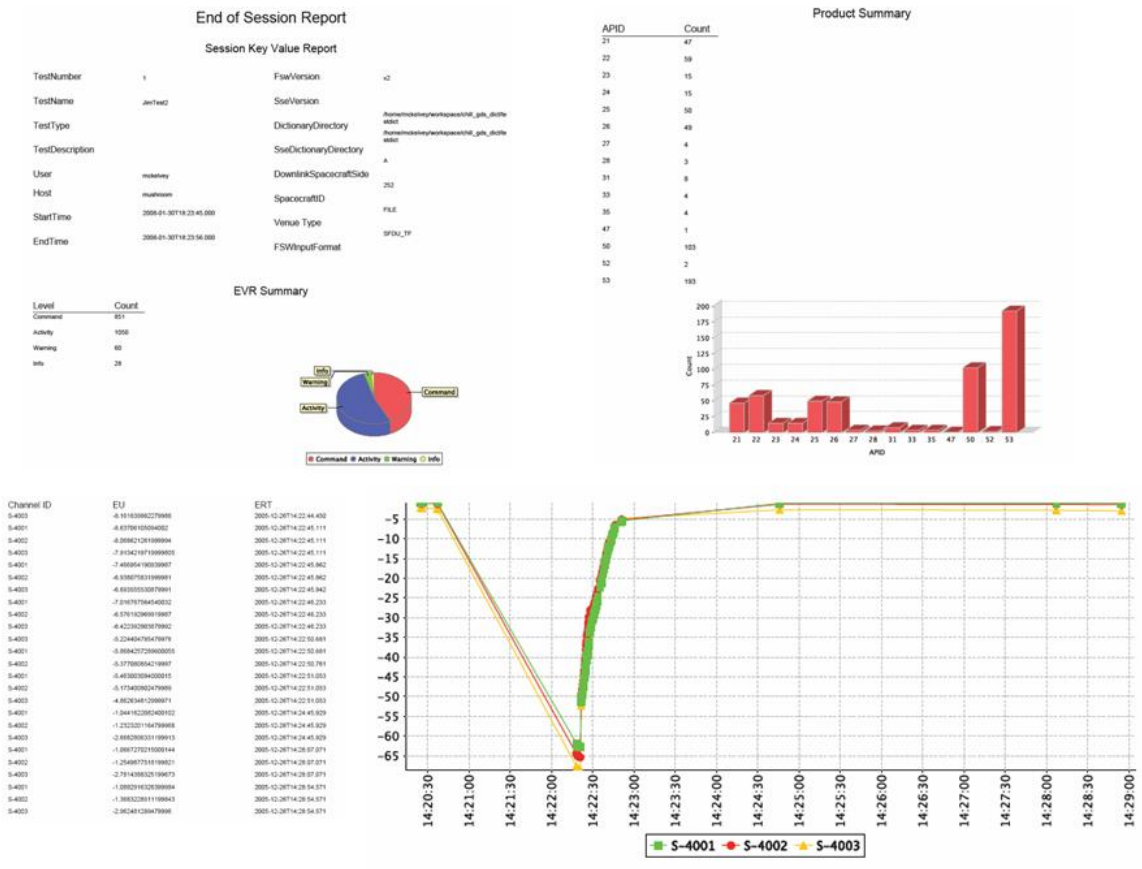

Figure 10. MPCS Example Report (End Of Session)

There are a number of similar efforts going on elsewhere to enhance GDS capabilities by supporting advanced high level displays and products. Among these is the concept of Situational Awareness Tools by GSFC (Dan Smith and Co), which focuses on providing users with advanced views of the current state of the GDS (MS-Project-like GANTT charts, Visio-like flow diagrams, etc). AMMOS is working with GSFC to investigate possible infusion of these tools, which appear to be completely compatible with MPCS and the JPL messaging standards.

V. Conclusion

This paper has described a next generation AMMOS product line called MPCS that has many powerful features, among them is the ability to create high level summaries and reports for mission users, produced by event triggers. These higher level products provide information, at a semantic level that most GDS users need. MPCS directly supports these products via a set of template systems (Velocity and JasperSoft), and the event driven design triggered by JMS messages. In the next year or so prior to MSL launch, MPCS will create advanced reports, similar to the MER examples, and even better – provide users with the ability to produce their own reports.

Acknowledgments

Many people have been working on MPCS over the last few years, and major acknowledgments should be made to Jesse Wright, Martha DeMore, Brent Nash, Lloyd Deforrest, Ashley Shamilian, Clark Williams, Mark Palm, and many others. A special thank you to Joe Kahr and the MSL Project, who provided early support for the effort (a "Perfect Storm").

The work described in this paper was conducted at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.