Open Source Next Generation Visualization Software for Interplanetary Missions

Jay Trimble
NASA Ames Research Center, Mountain View, CA, 94035, USA

and

George Rinker
Jet Propulsion Laboratory, Pasadena, CA 91109, USA

Mission control is evolving quickly, driven by the requirements of new missions, and enabled by modern computing capabilities. Distributed operations, access to data anywhere, data visualization for spacecraft analysis that spans multiple data sources, flexible reconfiguration to support multiple missions, and operator use cases, are driving the need for new capabilities. NASA's Advanced Multi-Mission Operations System (AMMOS), Ames Research Center (ARC) and the Jet Propulsion Laboratory (JPL) are collaborating to build a new generation of mission operations software for visualization, to enable mission control anywhere, on the desktop, tablet and phone. The software is built on an open source platform that is open for contributions (http://nasa.github.io/openmct).

I. Introduction

Built on the open source Open MCT (Mission Control Technologies) platform, available on GitHub at http://github.com/nasa/openmct, the Visualization for Telemetry Analysis (VISTA) client, deployed at JPL, and the Web Applications for Resource Prospector (WARP) client, deployed at ARC, bring mission control data visualization capability to the desktop, tablet and phones, using web browsers. The key features of the platform are data visualization, all of your data browseable and searchable in one integrated environment, user composition of displays, integration with multiple data sources, and modularity for customization to different mission requirements.

At JPL, the initial capabilities are targeted at remote access to telemetry, and composable dashboards to allow users to quickly build their own displays for rapid analysis of downlink data. At ARC, WARP will be used for distributed mission situational awareness across NASA centers, using multiple data types, such as telemetry, mission timelines, images, and rover surface traverse visualizations.

II. A New Open Source Platform

The decision to make the platform open was driven by the need to collaborate across organizations. Open source enables sharing and collaboration without ownership, as well as collaboration and contributions with organizations outside of NASA. Mission specific plugins may be open source or proprietary.

VISTA, deployed at JPL and WARP, deployed at ARC, are tailored to the requirements of each center. They share a common architecture and capabilities, but utilize different data sources and have mission-specific plugins. OpenMCT may be downloaded from GitHub at https://github.com/nasa/openmct.

The platform is built to deliver a set of core capabilities to missions and users:

A. Data first
   i. Users interact directly with their data
   ii. All data is viewable, browseable and searchable from one place
B. Users can compose displays from data objects using drag and drop
C. Objects from multiple domains may be assembled into layouts
D. Multiple data sources may be integrated and displayed
E. The software may be customized using plugins

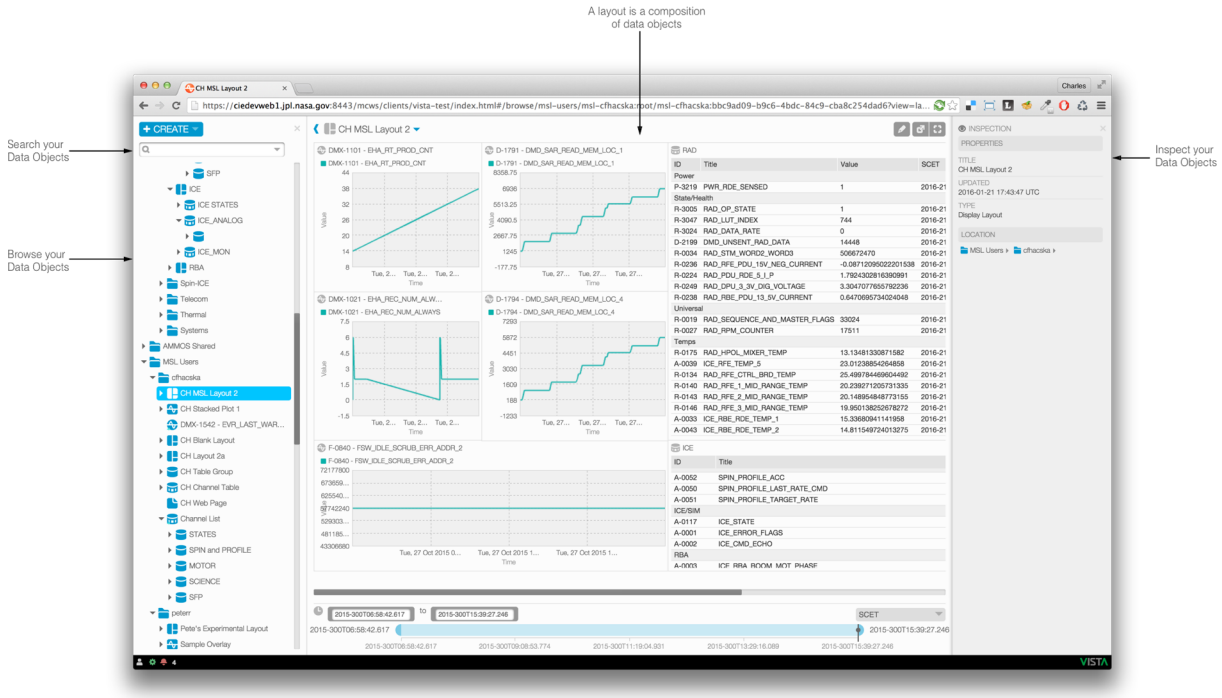

Figure 1 – This is a layout using MSL (Mars Science Laboratory) rover telemetry data, showing the object tree, from which the user may view, browse and search all of their data, the inspector, and the layout, which is a composition of objects.

Users interact with their data, not with applications. From the object tree, all data objects are viewable, browseable and searchable. Objects may be composed into layouts and inspected. Figure 1 shows the object tree, a composed layout and the inspector.

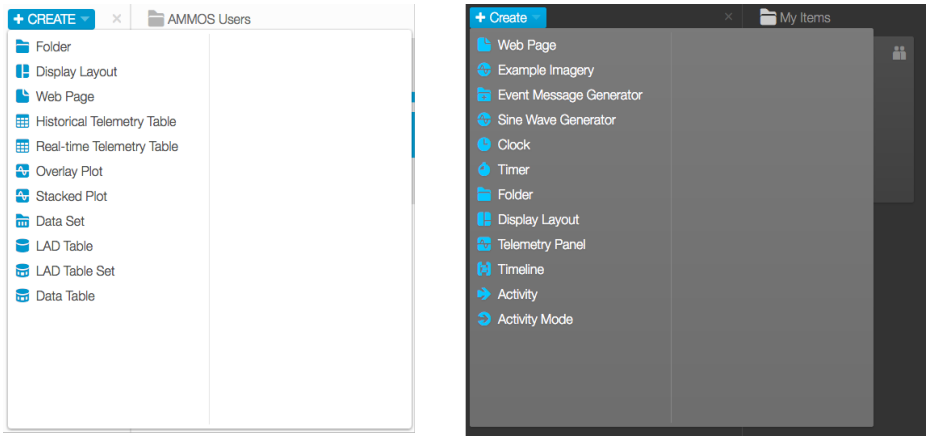

Figure 2 - The VISTA Create Menu at JPL (left) and the WARP Create Menu at ARC (right).

The create menu shows the list of objects that may be created and composed. Figure 2 shows two instantiations of the create menu. The version on the left is from VISTA at JPL, the version on the right is from WARP at ARC. Note that each deployment may be customized to use different objects. The JPL version is focused on telemetry and monitoring, the WARP version has a more diverse object set encompassing more mission operations areas.

Example compositions are shown in Figures 3 and 4. The mission "dashboard" (Figure 3) was built at JPL for the purpose of showing correlated telemetry data from the SMAP (Soil Moisture Active Passive) Flight Software. The ARC displays are used in support of the Resource Prospector (RP) mission, an in-situ resource utilization (ISRU) technology demonstration mission scheduled to launch in 2021. The RP displays support distributed operations

simulations at NASA ARC, Kennedy Space Center (KSC) and Johnson Space Center (JSC). Figure 4 shows some of the different object types that have been combined and displayed using WARP (and could be displayed and combined in VISTA). Figure 5 shows edit mode. This is where the user selects objects from the tree, and composes them in the layout.

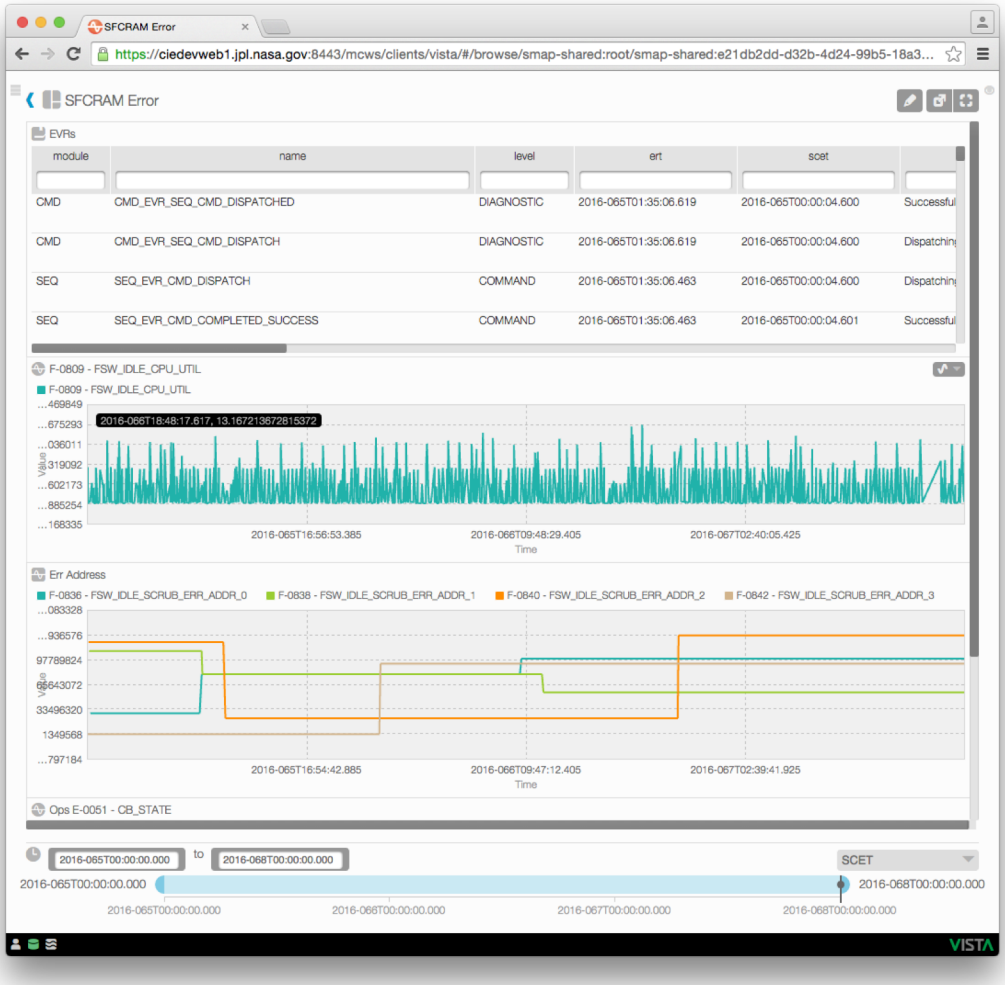

Figure 3 – JPL mission "dashboard" in VISTA showing correlated telemetry for analysis of the SMAP (Soil Moisture Active Passive) Flight Software.

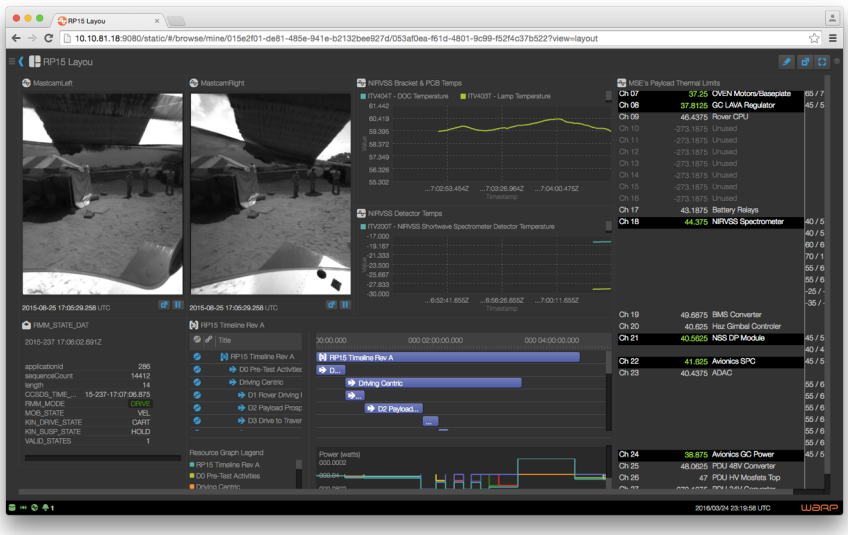

Figure 4 - WARP display combining timeline, activity, telemetry and image object types

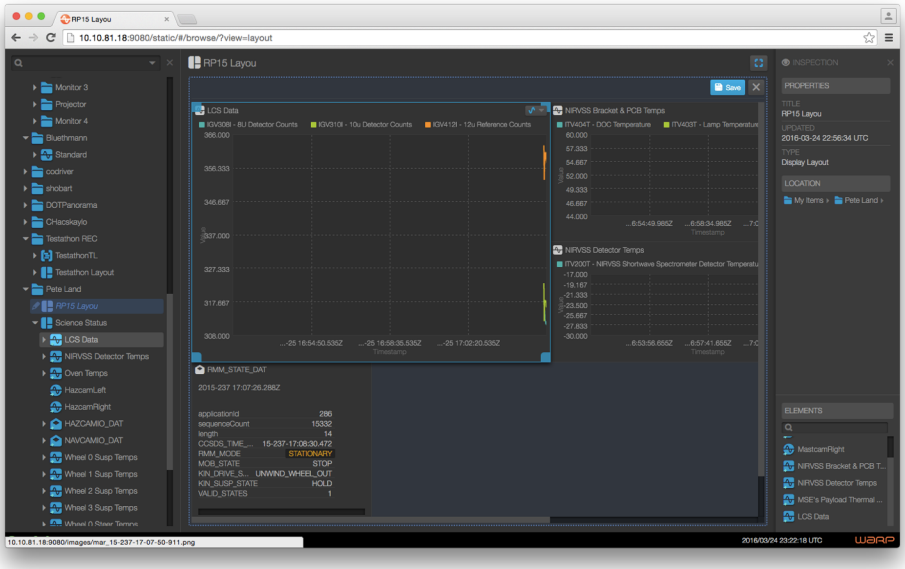

Figure 5 - WARP edit mode, used for composition. Note the subtle highlight in the layout and the save button. The user selects objects from the tree, using search or browse to find them, drops them into the layout, and drags to the desired position

The Open MCT platform allows users to compose displays by dragging and dropping objects into a display layout view. Unique compositions that integrate different data, or show the same data in different ways, can be defined for each mission operations role. Moreover, feedback from mission operators can be rapidly integrated, and changes incorporated without requiring expensive and time consuming changes to the underlying software. While a capability of the platform, the use of composition is a policy decision. For example real-time mission critical operations may happen on pre-composed screens that are not directly modifiable by the user during operations. By comparison, when performing post-facto data analysis, users of the system may be free to compose layouts suited to the task at hand.

Open MCT is a responsive web application that can be used on a variety of platforms, from desktops to smartphones. The only system requirement for using Open MCT is a modern web browser. Being web-based reduces maintenance overhead by removing the need to deploy updates directly to workstations. Because it uses a

responsive design, Open MCT can be used from mobile devices as well as desktops. Supporting mobile devices allows off-duty or on-call operators to maintain mission situational awareness outside of a mission control center.

III. Architecture

OpenMCT, written in Javascript, provides standardized telemetry Application Program Interfaces (API), an object model and matching persistence mechanism, and a tree model for organizing and composing objects to build displays.

The telemetry API's provide standardized patterns for interacting with real-time telemetry streams and historical telemetry stores, and provides a simple extension point for introducing new telemetry backends.

The key concept in Open MCT is the *domain object:* This is an entity that may be viewed and composed, such as a telemetry point, a timeline or a procedure. A domain object in Open MCT is expressed internally as the union of three concepts:

- An identifier. This is a string which uniquely identifies a given domain object within the running instance of the software (shared across all users of that instance.)
- A model. This is a JavaScript object which must be losslessly convertible to JSON (JavaScript Object Notation.) A domain object's model describes its common properties, such as its name, as well as its type-specific properties, such as telemetry metadata.
- Capabilities. These are dynamic behaviors associated with individual domain objects. For instance, objects with associated telemetry data have an associated "telemetry capability," which provides an interface for retrieving this data at run-time.

Expressing domain objects in this fashion allows for other application components, such as views, to be written in a loosely-coupled fashion. For instance, a view of telemetry data (such as the included plot view) becomes compatible with any domain object with a telemetry capability, allowing for variation in the telemetry-providing objects. These views are also able to safely ignore the domain objects for which they are not relevant; a Display Layout, for instance, has no telemetry capability and so a Plot view will not be made available. Other capabilities are more general; the "composition" capability, for instance, is exposed for objects which can contain other objects. This allows common composition-related features, such as the ability to drag and drop one object into another, to be implemented once and made available for all domain objects which have this capability.

A domain object's model, on the other hand, provides a flexible means of storing data and metadata associated with objects while also supporting common, simple solutions for serialization and persistence.

In the case of persistence, Open MCT provides a simple CRUD (create-read-update-delete) interface for storing and retrieving domain object models, typically serialized as JSON documents, to a persistence store. Multiple implementations of this interface are provided with Open MCT, persisting to browser local storage (useful during development) or to back-ends such as CouchDB and ElasticSearch. Plugins can introduce different implementations to allow Open MCT Web to sit atop different services; for instance, in VISTA domain object models are persisted to the Advanced Multi-Mission Operations System (AMMOS) Mission Control Web Service (MCWS) as JSON documents using MCWS's support for storing and retrieving opaque files, as shown in Figure 7.

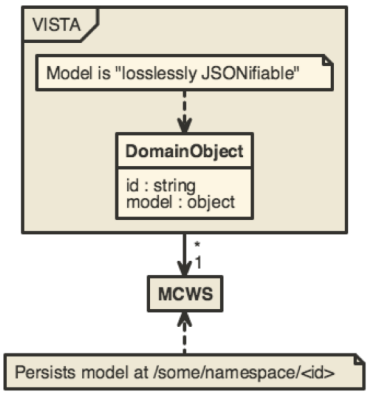

Figure 7 - VISTA persistence model; user-created objects are stored as JSON documents on MCWS.

This persistence interface is an example of the *ubiquitous extensibility* provided by Open MCT in support of its plugin mechanism. Components of Open MCT's architecture are expressed as *services* with clear, narrow responsibilities that fulfill specific roles within the architecture; these services can be extended using a few common patterns, illustrated in Figure 8:

- The Registry Pattern - This allows multiple things of the same type to be registered and then used. This is the simplest case, used by plugins that simply want to add new features, such as new types of domain objects, new views of domain objects, and so forth.
- The Composite Pattern - This allows multiple instances of

objects of the same type to be treated as one. Services which utilize this pattern are responsible for providing an implementation of a compositor which handles this one-to-many mapping. This pattern allows code which uses a service to be written as if there were one such service, when in fact there may be many. For instance, telemetry data may be requested across multiple sources from a single service. This pattern also ensures that the interface implemented for a new adapter for a certain service (such as a source of telemetry data) does not differ from the interface used by other components.

- The Decorator Pattern - This allows services to be augmented or enhanced without changing or specifying its underlying implementation.

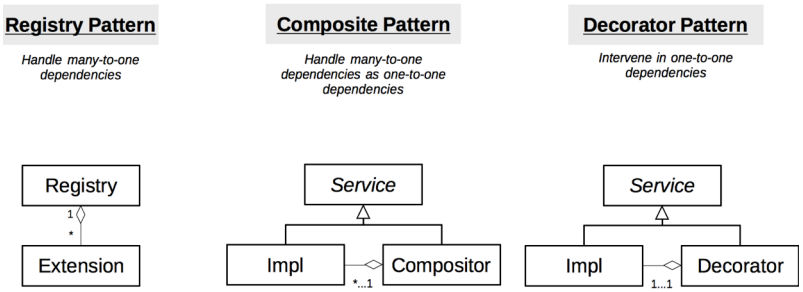

Figure 8 - Summary of commonly-used patterns which promote ubiquitous extensibility in Open MCT.

This plugin structure allows a wide variety of important components to be extended and adapted to meet the needs of the project:

- Adapters for different persistence stores may be introduced.
- Adapters for different telemetry sources (both realtime and historical) may be introduced.
- Mission-specific time systems may be defined.
- Custom status indicators may be added, typically to expose important system state (such as connection status for custom adapters) in a location which is always visible.
- Different sources for domain objects themselves may be introduced; for instance, a dictionary of available telemetry items is typically expressed as a hierarchy of domain objects exposed directly into the system, as an alternative to populating the persistence store with this same data.
- Additional views may be added, either to allow new types of information to be visualized, or to allow for existing types of information (such as telemetry) to be visualized in different ways.
- New types of domain objects may be added, typically along with new views for these objects. This is particularly useful to missions which have unique kinds of telemetry data, or unique operational concerns that they would like to integrate into an existing product.

VISTA utilizes Open MCT's plugin mechanism to provide sets of adapters specific to the JPL GDS (Ground Data System) environment (for multiple missions) which allow Open MCT to work with AMMOS Mission Data Processing and Control System (AMPCS), MCWS, Mission Planning and Sequencing Web (MPSWeb), JPL Analytics Cloud and other common ground data system elements out of the box. The extension points utilized by VISTA's various plugins are illustrated in Figure 9.

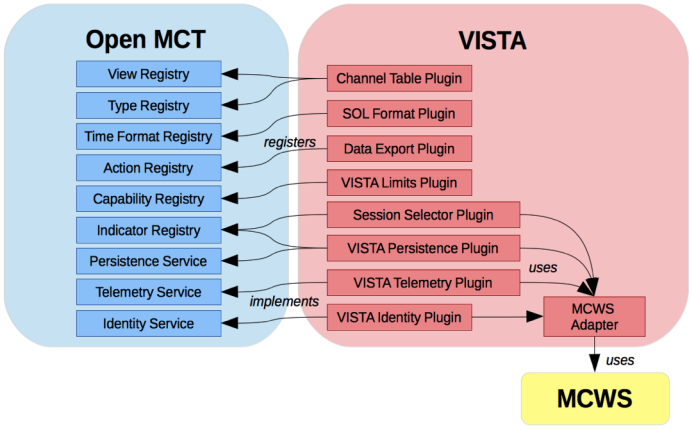

Figure 9 - Usage relationships between VISTA plugins, Open MCT extension points and MCWS

IV. VISTA at JPL

At JPL, the primary focus of VISTA is currently on telemetry displays. The current beta-version is running with data from the MSL and SMAP missions, for evaluation by mission controllers. Users are composing displays for analysis purposes using historical and realtime channelized telemetry data and event records. Frequently, users are leveraging these capabilities even in a beta-environment because the speed, composability, and extensibility of the user interface is unique and helps them perform high-level analysis quickly from a number of different mission essential perspectives. For example, Figure 10 shows VISTA displaying SMAP testbed information for four related channels over the same timespan, and Figure 11 shows VISTA displaying MSL operations information about historical trending for a set of important rover channels.

Users can configure VISTA to view multiple disparate datasets together, such as comparing data captured on a testbed on Earth with data from a spacecraft after launch. It also allows users to query the full life of mission datasets and helps them find repeat occurrences of anomalous events over long time-frames. Figure 12 provides an example of display composition with SMAP operations information with event records and channels over the same timespan.

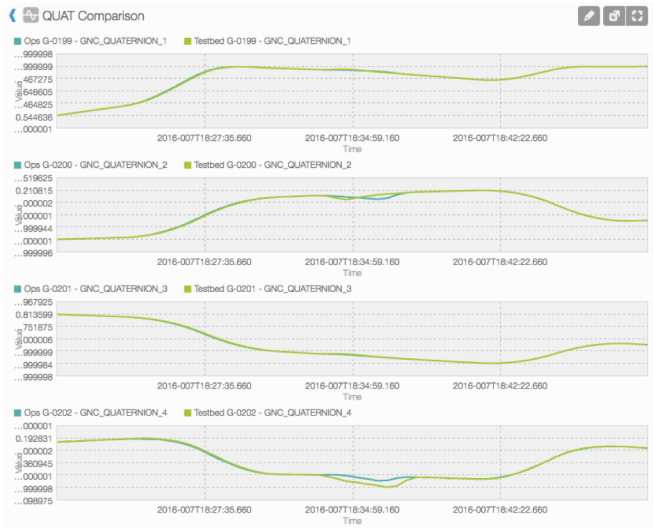

Figure 10 – VISTA display of SMAP testbed information for four related channels over the same timespan.

Operations teams use VISTA to correlate Event Records with channelized telemetry, and also to provide realtime awareness of spacecraft subsystems utilizing a Last Available Data (LAD) Query. LAD queries are optimized to provide the last value for a specified set of telemetry channels quickly so that they can view subsystem state and quickly locate out-of-bounds data. Additionally, LAD displays update with a realtime data stream to always show the latest state of the spacecraft. Figure 13 shows a VISTA display for the SMAP spacecraft with a set of related channels some of which are in a yellow alarm state.

Missions are currently considering VISTA for extended uses, such as timeline and image display which are already implemented in WARP at ARC for RP.

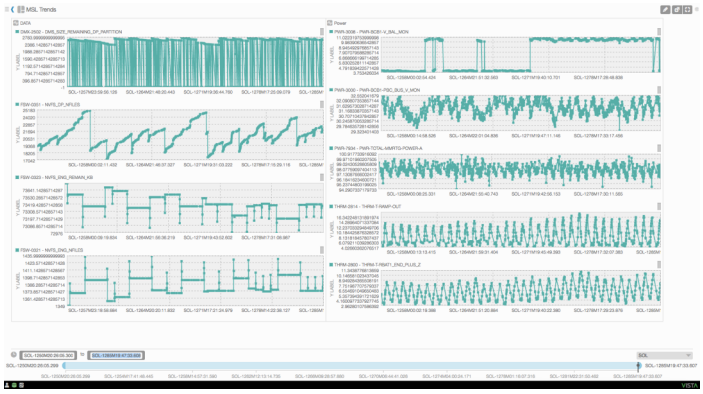

Figure 11 - VISTA display showing trending information for important MSL rover channels.

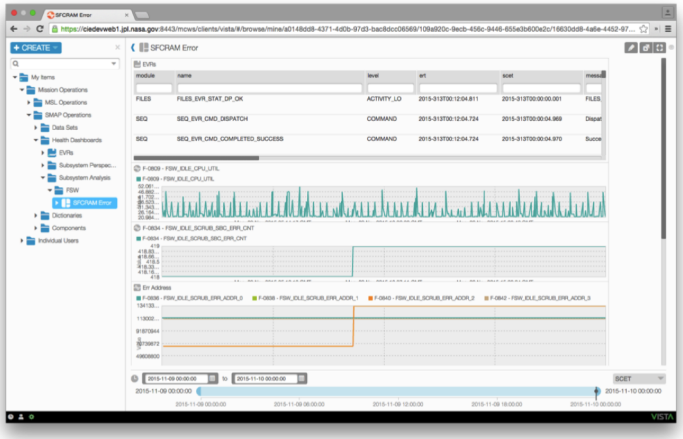

Figure 12 - VISTA composition showing event records correlated with channelized telemetry.

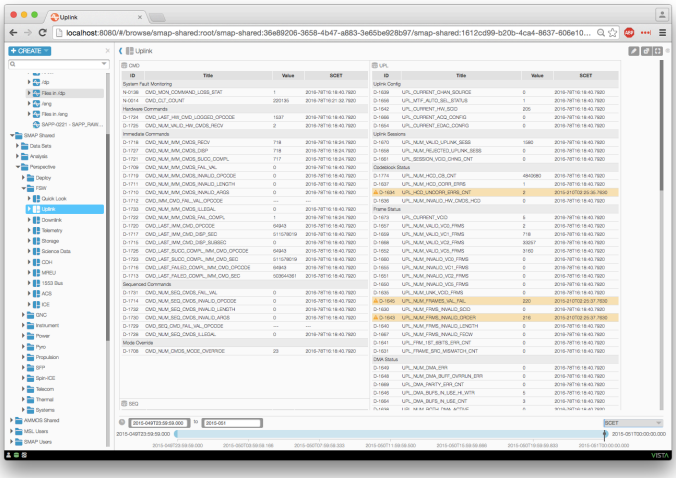

Figure 13 - VISTA LAD display showing subsystem status and highlighting alarm states for the SMAP spacecraf

VI. WARP at ARC

WARP will support the RP Lunar Rover, scheduled to launch in 2021 to prospect for volatiles at the South Pole. WARP is already in use to support early simulations for design, and to support flight system and payload development. The use cases identified for WARP to date are:

- Distributed situational awareness
  - Timelines
  - Telemetry
  - Procedures
  - Traverse visualization
  - Imagery
  - Control center cams
- Access to data anywhere for off shift operators
- Continuously connected mission operations

In the Summer of 2015, the RP mission conducted a Distributed Operations Test (DOT) in which a prototype rover, payload, and operations system tested key concepts necessary to conduct the mission. Using prototype rover and payload hardware operating at JSC, and a distributed mission team operating at ARC, KSC and JSC, the team executed key elements of the mission timeline, validating the operational concepts.

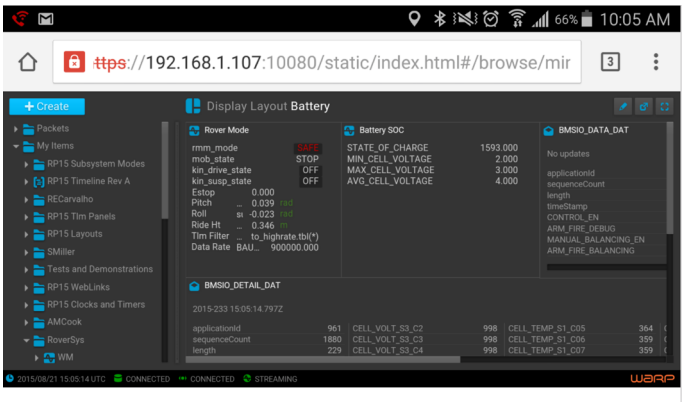

Figure 14 - WARP on a mobile phone, used for monitoring rover system health in the field

Figure 15 - The RP-15 Prototype Rover and Payload at JSC.

WARP, running in web browsers on desktop computers, was used for telemetry display, activity timeline display, and image display for decision support for rover driving. WARP was tested on iPads at ARC. At JSC, where the rover was operated in the field, WARP on a mobile phone augmented WARP running on a laptop, to monitor rover systems. The prototype rover-payload was operated at JSC, so the capability to run WARP on a phone was useful to rover systems, who had to be able to monitor data from the field.

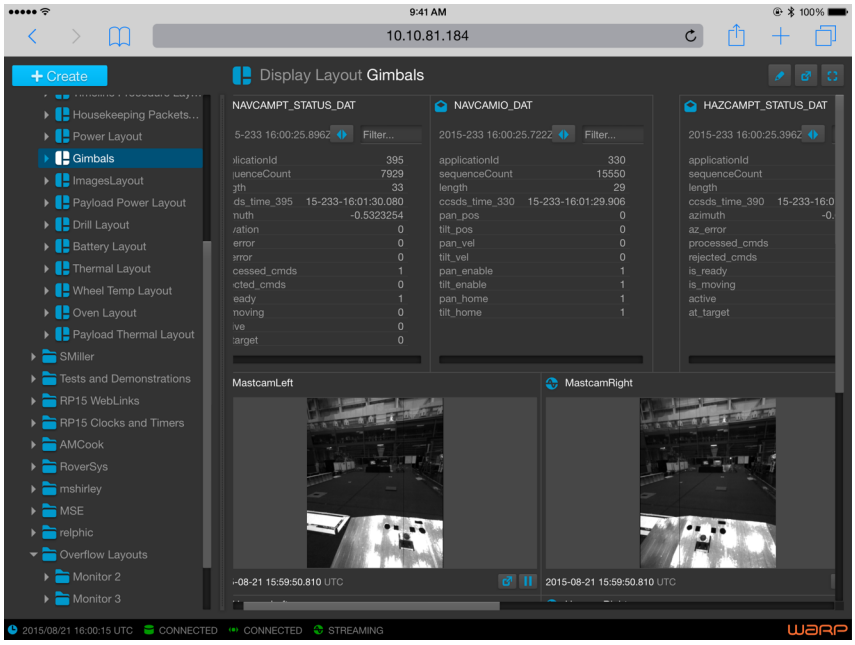

Figure 16 - WARP on iPad during the distributed operations test.

VI. Conclusion

We have developed a flexible platform for mission control visualization that is adaptable for developers through plugins and for users through composition. Initial use on MSL and SMAP at JPL, and RP at ARC, has seen adoption and positive response from users. The list of new feature requests is long. With ongoing development, we will test the viability of bringing functionality that previously existed in multiple applications into one integrated user environment, on multiple devices. The open source platform (https://github.com/nasa/openmct) provides a foundation that enables collaboration through shared ownership, and opens up the possibility of contributions and collaborations outside of NASA, with an open source community that is just beginning.

Acknowledgments

Thanks to the development team, and those who also made contributions:

1) Victor Woeltjen, Pete Richards, Andrew Henry for contributions to content and many proof reads

2) The Open MCT/VISTA/WARP Design and Development team at ARC – Charles Hacskaylo, Andrew Henry, Pete Richards, Victor Woeltjen;

3) The JPL core team for VISTA and MCWS development and testing – Dan Allard, Hayk Arutyunyan, Luis Campos Bravo, Dave Santo, Ashley Shamilian, Nicole Witek

4) The extended team at JPL - Pearl Haw, Frank Hy, Dan Isla, J.P. Pan, Farzad Saadat, Rachel Witte, and Lavin Zhang.

References

1. Curbow, D. and Dykstra-Erickson, E., Designing the OpenDoc Human Interface, https://pdfs.semanticscholar.org/8539/69d8b03b725139979fa09ed6bb82b1f60616.pdf
2. Johnson, J. Roberts, T, Verplank. W, Smith, D. Irby, C. Beard, M. Mackey, K The Xerox Star: A Retrospective, http://members.dcn.org/dwnelson/XeroxStarRetrospective.html
3. The Democratization of Mission Control, Empowering Users, Jay Trimble, Tom Dayton, Alan Crocker, CHI 2013.
4. Squeak/Smalltalk, http://squeak.org
5. Trimble, J., Reconfigurable Software for Mission Operations, AIAA Space Ops 2014
6. Trimble, J., Open Source Software for Mission Operations – Technology, Licensing and Community, AIAA SpaceOps 2014
7. Trimble, J., Dayton T. and Quinol, M., Putting the Users in Charge: A Collaborative, User Composable Interface for NASA's Mission Control, Hawaii Internationl Conference on System Sciences, 2012
8. Trimble, J., Crocker, A., Reinventing User Applications for Mission Control, AIAA SpaceOps 2010
9. Trimble, J., Crocker A., A Flexible Evolvable Architecture for Constellation Mission Systems User Applications, AIAA SpaceOps 2008
10. Trimble, J., Walton, J., Saddler, H., Mission Control Technologies: A New Way of Designing and Evolving Mission Systems, AIAA SpaceOps 2006