Reconfigurable Software for Mission Operations

Jay Trimble
NASA Ames Research Center, Moffett Field, Mountain View CA 94035

We developed software that provides flexibility to mission organizations through modularity and composability. Modularity enables removal and addition of functionality through the installation of plug-ins. Composability enables users to assemble software from pre-built reusable objects, thus reducing or eliminating the "walls" associated with traditional application architectures and enabling unique combinations of functionality. We have used composable objects to reduce display build time, create workflows, and build scenarios to test concepts for lunar roving operations. The software is open source, and may be downloaded from https://github.com/nasa/mct.

I. Introduction

To ensure mission safety, mission operations software requires strict change control processes following the start of use for mission training and operations. The same processes that keep missions safe can hinder the effectiveness of the software for operations. This is because software is built based on requirements that are developed years ahead of a mission. The software is released and used during training and simulations in a mission-like setting for the first time. Feature needs and display configuration requirements that were not known during requirements definition and review become apparent only then. Requests for change during simulations and mission operations are difficult to meet given the strict governance model. When the software cannot be changed the result is operational workarounds that the mission team must implement. This can result in significant operational overhead for a mission.

We have created a software technology that gives missions a greater degree of flexibility than has been possible previously. The software, Open Mission Control Technologies (Open MCT), combines the capability to compose software using drag and drop, with a policy manager that allows organizations to control flexibility. It is freely available from GitHub at https://github.com/nasa/mct. The composition capability allows for rapid change and reconfiguration. The policy manager allows for control of the flexibility provided by the composition capability. An organization can control which users or user groups have permissions to create, modify, or view displays.

The composition capability enables the creation of cross-discipline composite displays. This frees the mission team from traditional discipline-based roles, and allows for any parameter or group of parameters from any discipline to be put together and reused in composite displays. An example is a flight director's display that shows a timeline, telemetry, procedures, and commands. These objects may be combined in non-traditional ways. A flight log can be dropped into a timeline to show time-based notes compared to actual telemetry values. Telemetry may be embedded in procedures or flight rules. All of this may be reconfigured by drag and drop, based on project policies.

The software has been used in multiple domains. It was certified for display of telemetry from the International Space Station (ISS) at the Johnson Space Center. It has been tested at the Jet Propulsion Lab (JPL) for the display of data from the Mars Science Lab (MSL—Curiosity) and as a testbed for the next generation display of time based data. At NASA Ames the IRIS Mission is using MCT to view archive telemetry. ISS biosciences missions are using MCT to display data. An Advanced Exploration Systems (AES) Mission is using MCT to model operations on the Lunar Surface. Open source users ranging from commercial space companies to hobbyists have experiemented with Open MCT.

II. Aspects of Reconfigurable Software

The Open MCT software is, as the name suggests, open source, and can be found at https://github.com/nasa/mct. There are two fundamental aspects of Open MCT that provide reconfigurability: modularity and composability. Modularity provides the capability to add and remove functionality using plug-ins. Plug-ins are implemented as OSGi bundles (http://www.osgi.org/Main/HomePage). Composability refers to the capability of users to assemble

software from pre-supplied parts, which we call “objects.” The user is presented with a user environment that is an object container from which they may assemble collections of objects on a canvas to create operational displays.

Modularity

A. Modularity for Developers

Developers utilize modularity to structure the code for development and testing. The platform plug-in structure, shown in Figure 1, was built to enable separation of functions from each other, to improve platform stability, testing, and evolution. This proved itself during development when we had issues with plot performance. We were able to troubleshoot and try different solutions for plotting by installing and removing plug-ins. This enabled the team to isolate issues and solve problems faster than a monolithic architecture would have. Figures 2 and 3 show a plot test setup using different plot packages.

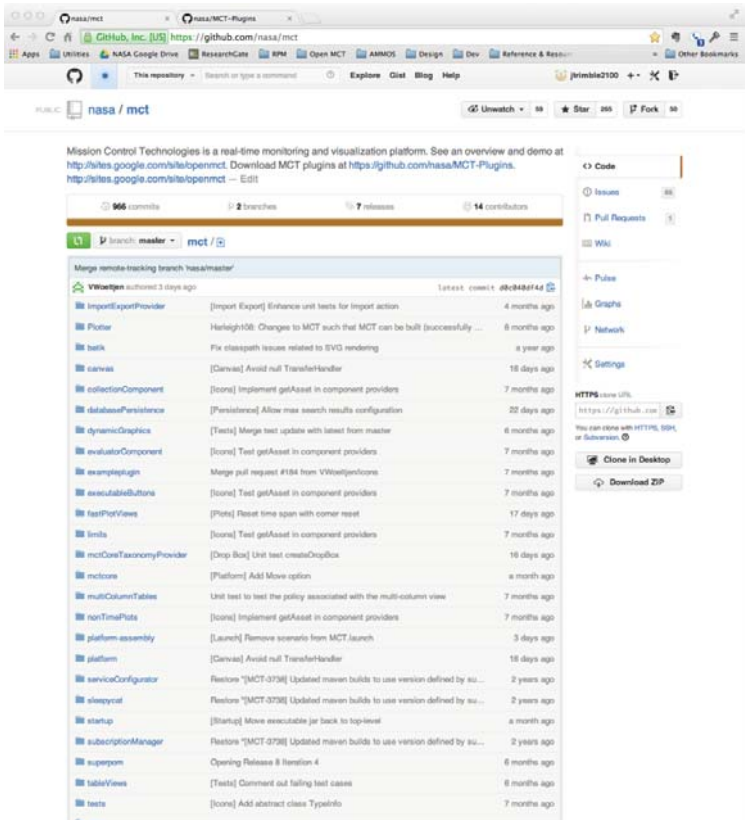

Figure 1. The Open MCT Platform Plug Ins - https://github.com/nasa/mct

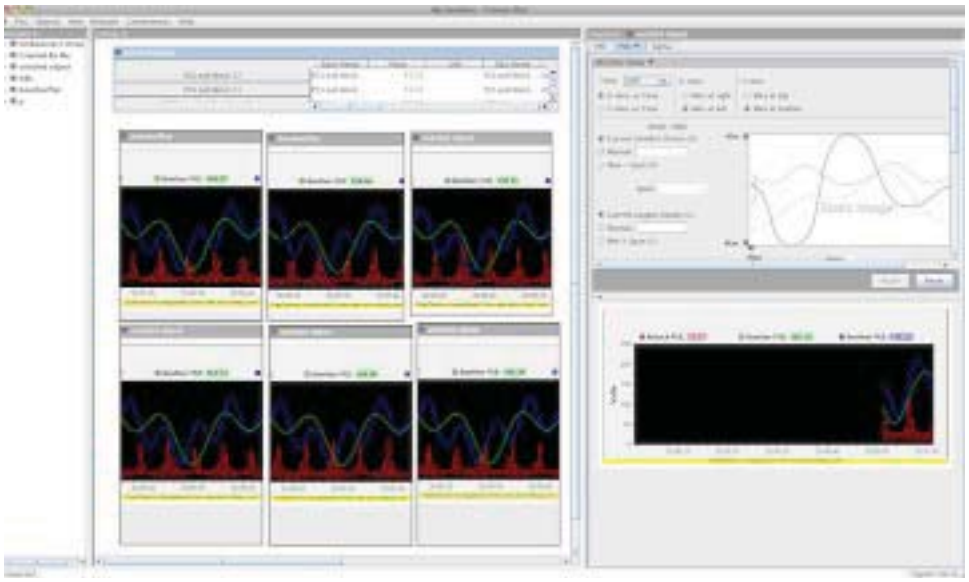

Figure 2. Plot Package being used for troubleshooting. Different plot plugins are displayed as panels, so their behaviors can be compared.

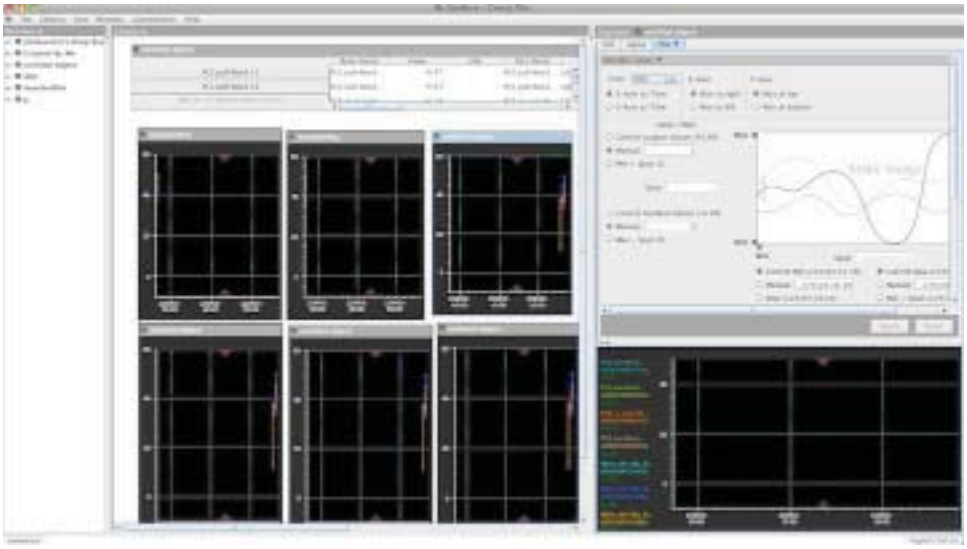

Figure 3. Baseline plot package in the same config as the troubleshooting package. The same plot plugin is displayed in multiple panels, the panels showing different data, or the same data but with different scalings or other adjustments.

B. Modularity for Users and User Organizations

For users and multi-mission organizations, modularity provides a path to add and remove features at a functional level, thus presenting users with what they need but not more. The primary benefit is the reduction or elmination of code and feature bloat. With monolithic architectures, when features accumulated in multi-mission software everyone got those features whether they wanted them or not. With modularity the addition and removal of features may be done by adding and removing plug-ins from a folder. The plug-ins folder is shown in Figure 6. The open source plug-ins page on GitHub is shown in Figure 4 (https://github.com/nasa/MCT-Plugins). Figure 5 shows the object creation menu, showing the objects that users may create and compose. The objects in this menu vary with the installed plug-ins. Objects in the menu correspond to one or more plug-ins in the directory.

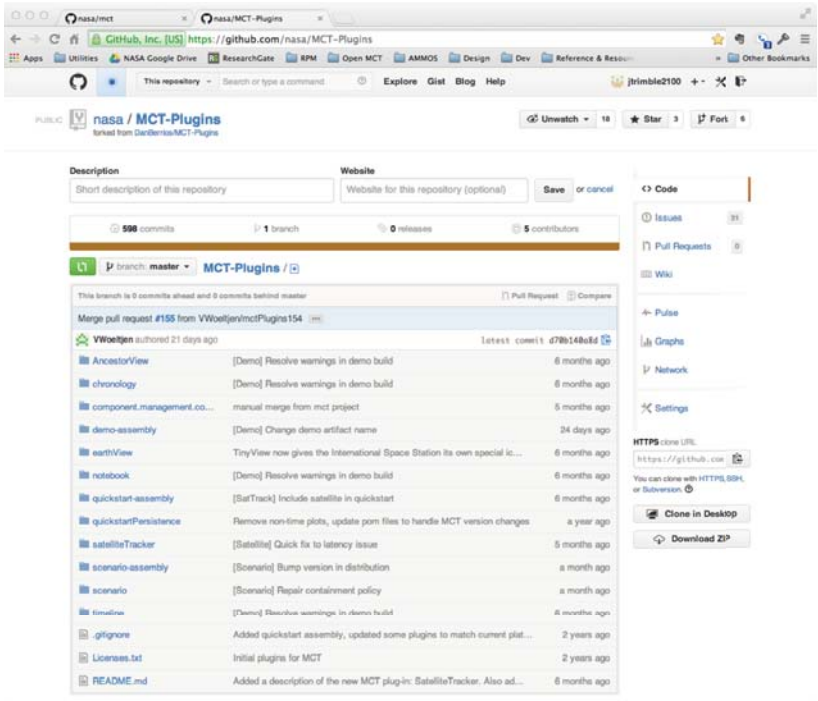

Figure 4. Open MCT Plug-Ins, https://github.com/nasa/MCT-Plugins

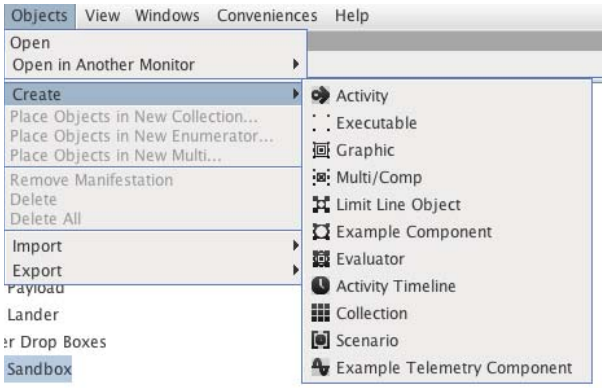

Figure 5. User can create object instances. The installed plug-ins determine which objects are available to users

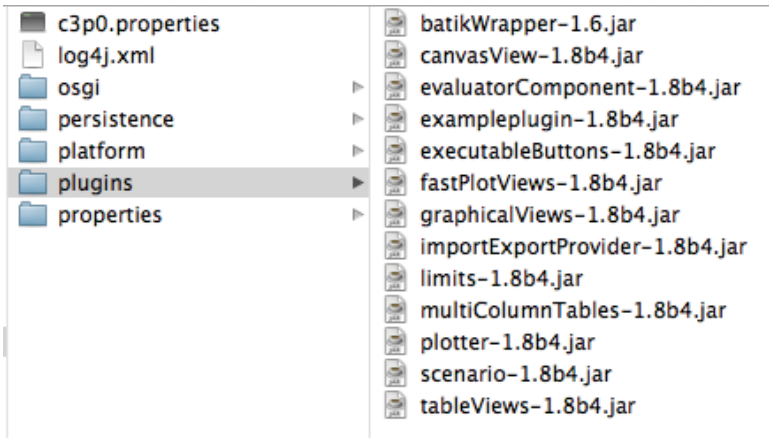

Figure 6. The plug-ins directory determines the objects available for users

Composability

Composition refers to the ability of users to assemble software from objects, and to lay out and view those objects in different ways. Each object contains core attributes that define the object's properties. For example, a telemetry point represents a specific value from a channel. The visual representation of objects may differ. MCT objects are "live" so they may be viewed in different ways. Figure 7 shows multiple views of an ISS telemetry object. *This is the first property of composition—the ability to view the same thing in different ways.*

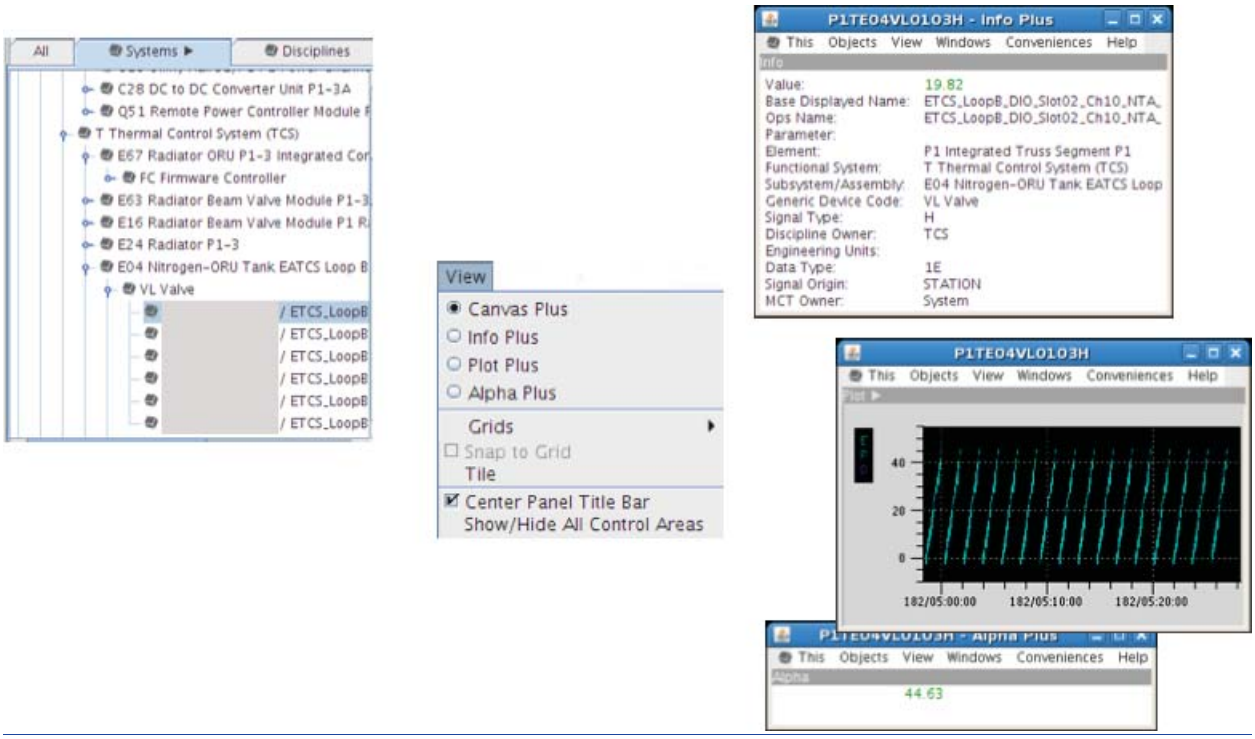

Figure 7. The same telemetry in different views

The next property of composition is the ability to assemble collections of objects from what would traditionally be different applications. Objects are assembled in collections, a collection being an empty container. Figure 8 shows telemetry in a collection. To the left is the directory, which is a listing of all objects in the collection. The center is the canvas area, in which objects may be laid out, arranged, and viewed. The view may be changed live at

any time. On the right is the inspector. Clicking on an object brings it up in the inspector. Views may be changed live in the inspector.

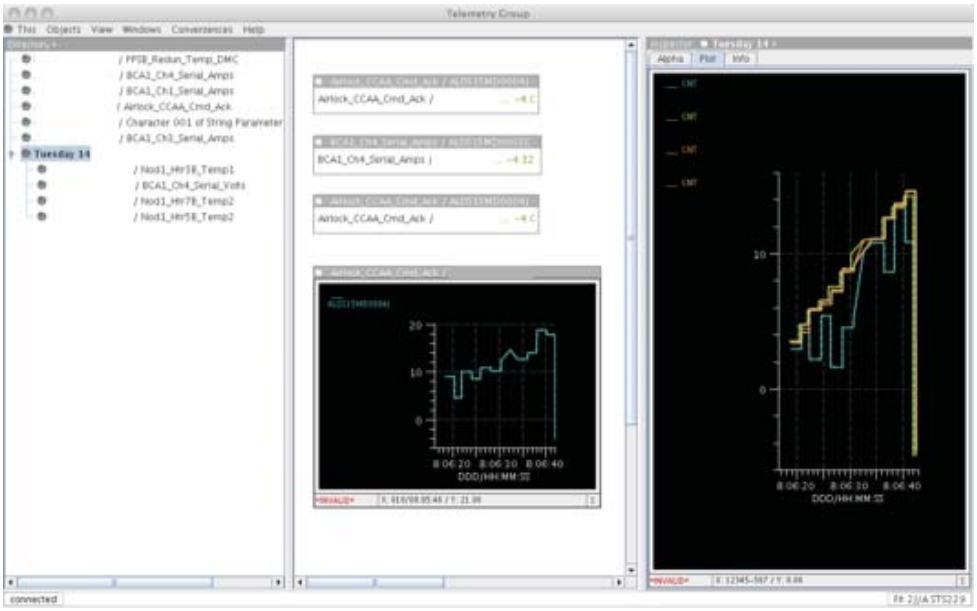

Figure 8. Telemetry in a collection shown in canvas view with an inspector

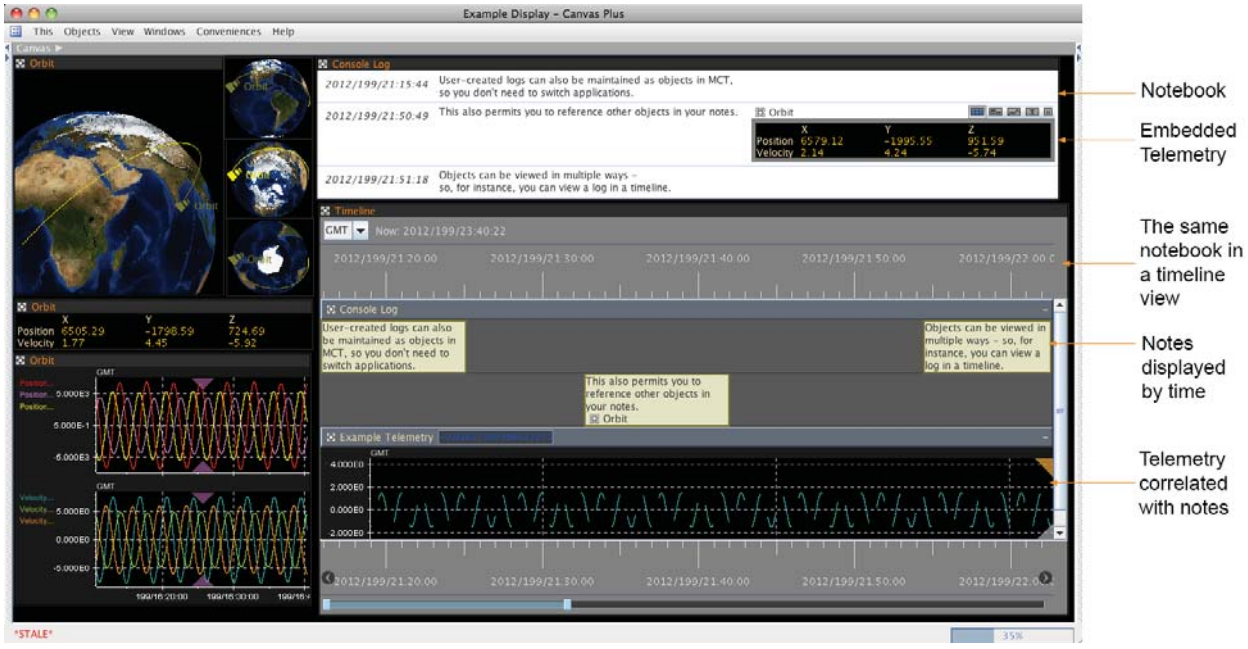

Figure 9. A multi-domain composition showing plots, timelines, logs, orbits, and data correlation over time

Figure 9 combines objects from different domains and shows combinations not possible with traditional application architectures. At the top is a console log. It is composed of text entries time tagged to correlate with mission time. Each entry allows the user to embed other objects. In this case telemetry has been embedded into one

of the entries. The view on top is a notebook view. The view on the bottom is a timeline view of the same object. In this view the user can see the telemetry in time with the notes.

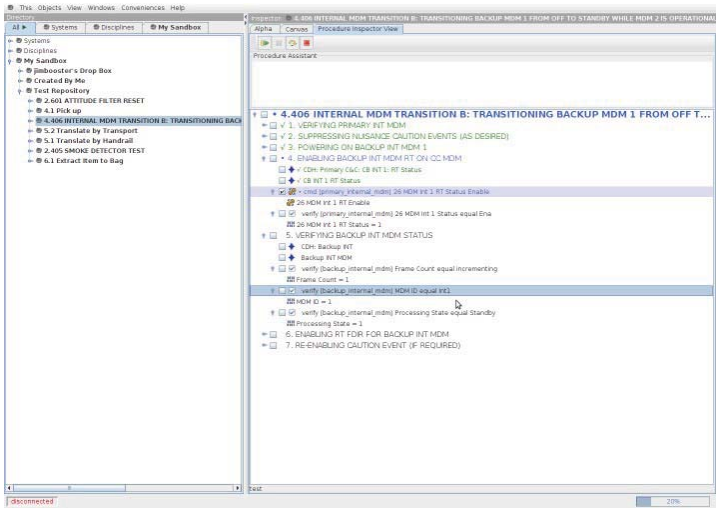

Figure 10. Procedures

Figure 10 illustrates another object type, procedures. Each step is a composable object.

Benefits of User Composition – What We Know So Far

We get different anecdotal responses from customers in regards to composition. Some respond positively, others suggest that they want to build their displays, then fly their spacecraft and not change anything. Below are examples of some of the benefits provided by composition to date.

C. Metrics for building displays

One of our customers performed a comparison of the time and number of steps to build a new display using MCT compared to their existing (legacy) system. Figure 11 illustrates the results. In short, the process time was reduced by 90% and the number of steps was reduced by 60%, with an 80% reduction in manual entry. The displays being built for test purposes were replications of the existing telemetry displays for the International Space Station. This data represents the efforts of users who were highly familiar with the product and had participated in the design.

Metrics

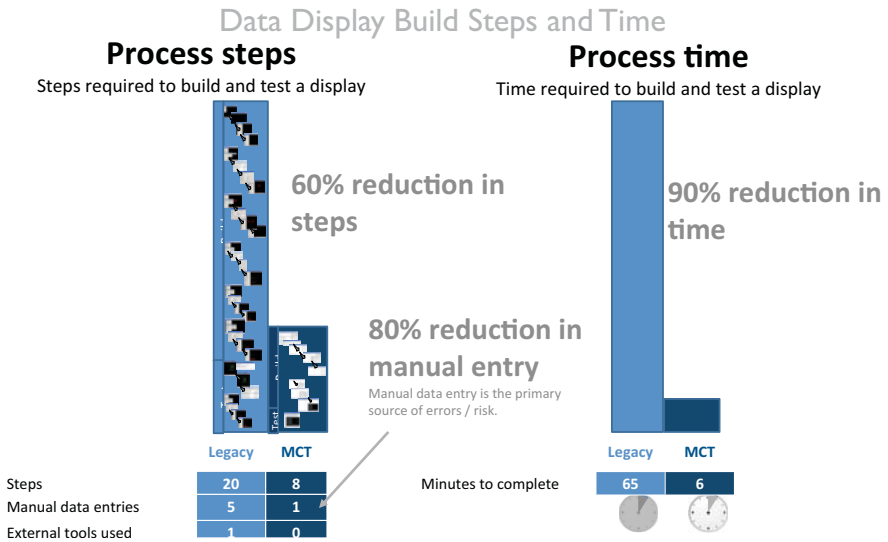

Figure 11. Time and process reduction in display build time measured by one customer

D. Search Complements Build

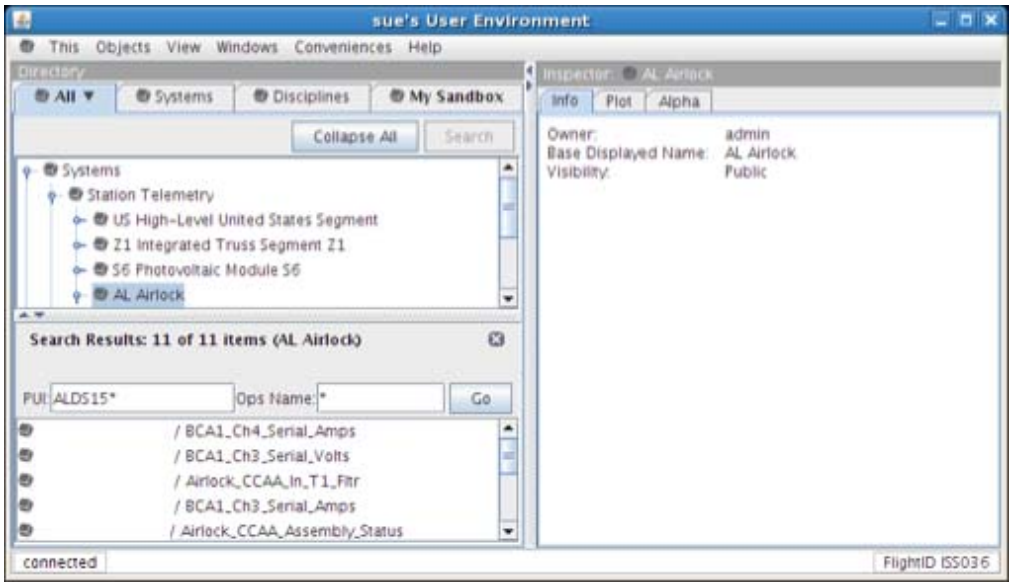

Figure 12. Searching by ISS PUI or Ops Name

Current mission systems typically build displays to show all parameters that mission users want to see. For complex systems this can involve building displays to show thousands of parameters. With composability and search, a different strategy may be employed. Since all parameters are available and easily found, we believe it is not necessary to have pre-built displays for all parameters. Instead, a mission can build displays for the most commonly viewed parameters. If it becomes necessary to view other parameters they may be found using search or browse and quickly viewed *or composed into new collections*. Those collections may be shared for others to see.

E. Reversing the parameter association paradigm

Traditional applications associate a display element with a specific parameter. Reuse of that element requires re-wiring the parameter association with each use. MCT reverses this paradigm, by creating an object with associated properties. The association need only be done once and the object may be reused. This is illustrated in Figure 13.

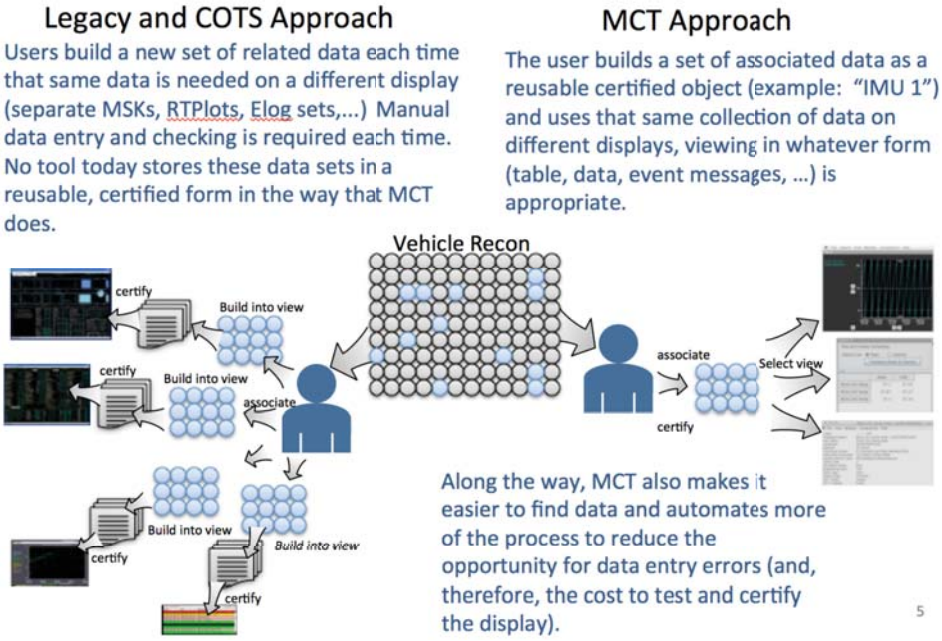

Figure 13. Object association with parameters need only be done once

F. Designing Scenarios for a Lunar Roving Mission

We present a brief look at using Open MCT's composition capability to develop early operational scenarios and operational units to support the design of the Resource Prospector Mission (RPM)—a lunar roving in-situ resource utilization mission scheduled for a 2019 launch. In support of the mission concept review (MCR) and early design, we are using Open MCT to answer basic questions:

- What is the total elapsed time on the surface to reach minimum and full mission success criteria?
- What are the power consumption and data usage profiles?
- What are the units of operation for the payload, lander, and rover? What is the optimal level of granularity to represent operations?

To do this, we built plug-ins that added three object types to Open MCT: A *scenario* is a master container into which multiple timelines may be added. The scenario time syncs the contained timelines with each other. A *timeline* is a container of activities and decisions. An *activity* has attributes of name, time, power, and data usage. Activities may be created as types that can be reused.

Figure 14 shows a part of the RPM cruise scenario. The directory on the left shows all of the objects contained in the scenario. The scenario contains two timelines—one for the lander and one for the payload. Either or both of those timelines may be viewed and edited outside of the scenario container (Figure 15). Each timeline contains activities and sub-activities which, together, form a grouping of operational units that may be moved as a block and, in effect, constitute a decisional block for reactive near-real-time operations on the lunar surface. The benefit of composability is that these units of operation need not be decided in advance. The number of timelines in a scenario, the activities and sub-activities, the groupings, the activity types--all of these may be built and composed by users. The core structure may be changed based on lessons learned--all using composition, with no programming.

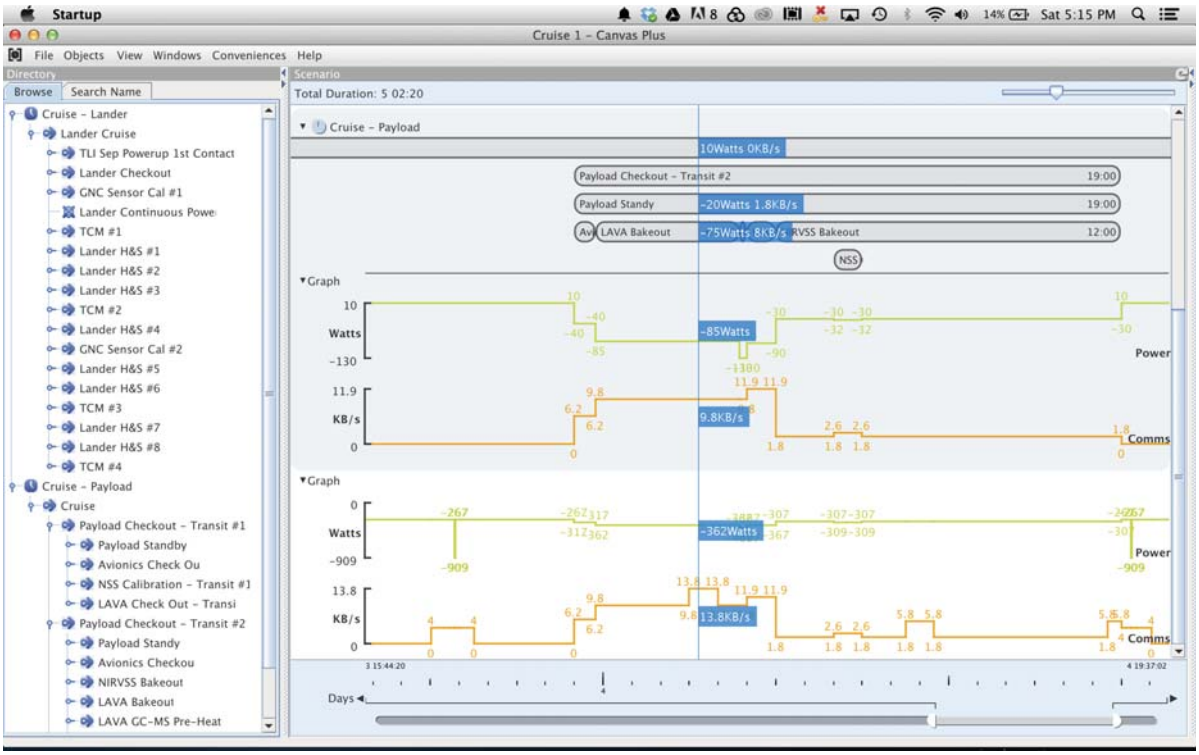

Figure 14. RPM Cruise Scenario

Other noteworthy items in this scenario: Because each element--the scenario container, the timelines, the activities—all are objects, with consistent behavior, they may be inspected and acted on in or out of the scenario container. To open a timeline on its own, the user simply double clicks on the object icon for the timeline. Each timeline has its own graph showing power and data versus time. The graph may be hidden. The scenario has a graph showing total power and data usage for all of the timelines. The user may display power and data usage by activity, simply by dragging their cursor along the bottom of the timeline. This is the line shown in Figure 14.

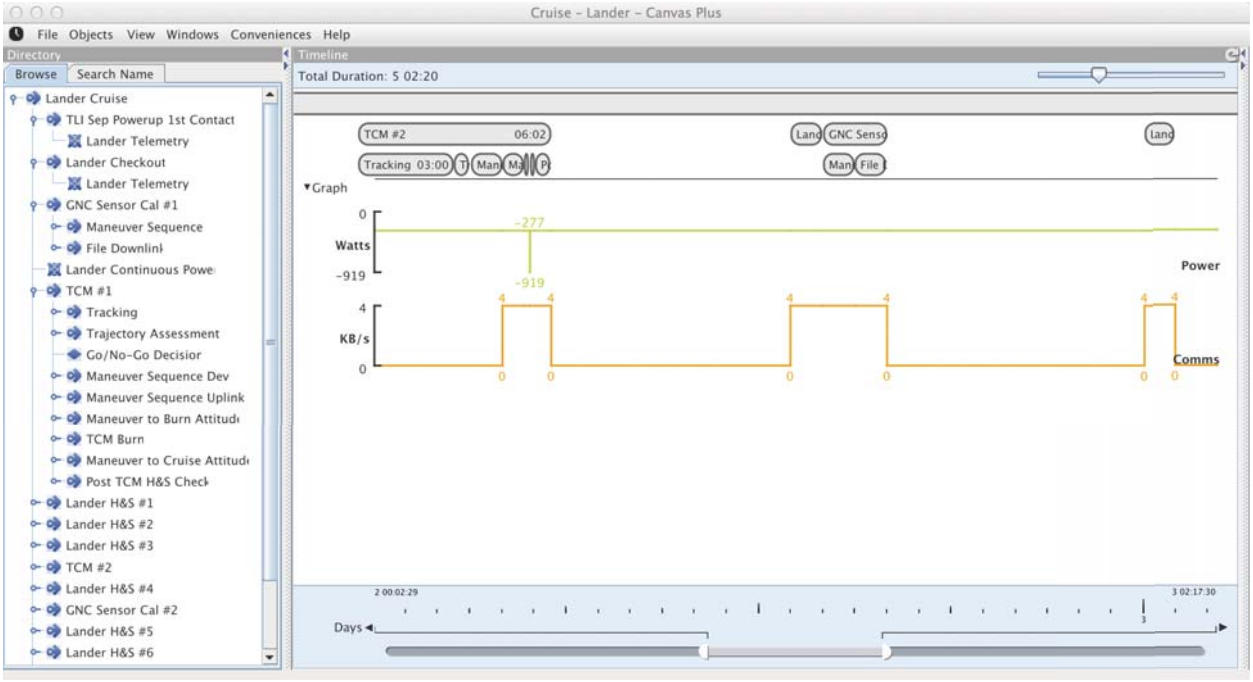

Figure 15. The lander timeline from the cruise scenario displays outside the scenario

Figure 14 shows the lander timeline displayed outside of the scenario. We have developed the timelines by element (lander, payload, rover) and then put them into the scenarios.

For RPM scenario creation we have a used three-tiered structure. My Sandbox is an area for individual users to experiment with scenarios, visible only to them. The Shared area is set up to share and develop scenarios. The Mission area is for approved scenarios that have been reviewed and agreed on. This structure was not predetermined and may be changed any time. We used collections and permissions to set up a workflow.

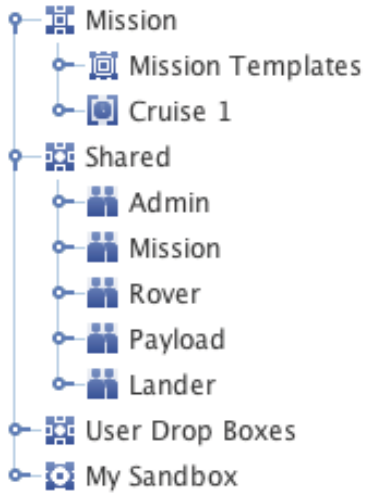

Figure 16. Workflow Structure in Collections

III. Conclusion

Modularity and composability provide unprecendented flexibility to missions, users, and multi-mission organizations. Modularity has enabled structured development and testing, and rapid reconfiguration to suit different mission and organizational needs. Composability has enabled rapid display creation, has reduced the number of displays that we need, and has provided a structure for mission scenario design that minimized up front design and enables a broad exploration of the possible design space.

References

[1] Trimble, J., Walton, J., and Saddler, H. "Mission Control Technologies: A New Way of Designing and Evolving Mission Systems." Spaceops 06

[2] Trimble, J. and Crocker, A. "A Flexible Evolvable Architecture for Constellation Mission Systems User Applications." , Spaceops 08

[3] Trimble, J. and Crocker, A. "Reinventing User Applications for Mission Control,." Spaceops 10

[4] Trimble, J., Dayton, T., and Quinol, M. "Putting the Users in Charge: A Collaborative, User Composable Interface for NASA's Mission Control." Hawaii International Conference on System Sciences

Acknowledgements

Victor Woeltjen, Charles Hacskaylo, Sue Blumenberg, Tom Dayton, Madelyn Quinol, Sarah Hobart, Alan Crocker, Kevin Jennings, Chris Webster, Nija Shi, Joshua Keely, Chloe Abraczinskas