A Flexible Evolvable Architecture for Constellation Mission Systems User Applications

Jay Trimble[1]
NASA Ames Research Center, Mountain View, CA, 94035, USA

Alan Crocker[2]
NASA Johnson Space Center, Houston, TX, 77058, USA

While simulating a complex set of repair tasks to be performed by EVA crewmembers on an upcoming mission, flight controllers and astronauts determine that the repair will take much longer than originally anticipated. All equipment in the vicinity of the worksite must be powered off to maintain a safe environment for the astronauts. Because heater power will be unavailable, several critical components will now be at risk of freezing and permanent damage. If an impending thermal violation is detected, Mission Control will have very limited time to react. Therefore, flight controllers must not only modify their procedures to account for these risks, they must also incorporate into their displays outputs from thermal models, alternate temperature measurements, new alarm limits, and emergency power-on commands to enable the detection and response to freezing conditions.

Current software for mission control systems makes scenarios like this difficult to address. Given the time frame for modifying software, operations teams are left with labor-intensive operational workarounds as their only options. NASA Ames Research Center (ARC) and NASA Johnson Space Center (JSC) are collaborating on the development of a flexible software system for mission operations that will enable greater user flexibility than has been available to date. Using composable software, end users in the scenario described above could recompose procedures and command and control displays to allow flight controllers to monitor temperature measurements, identify time-critical conditions, and execute the procedures required to respond to these conditions before flight hardware is permanently damaged.

I. The Fundamental Idea

The fundamental idea behind the Mission Control Technologies (MCT) project is to build software from small pieces that can be assembled by end users to create integrated functionality. Applications are built as compositions of "live objects" that can be combined in different ways for different users and missions as required, in contrast to the more traditional software development method of pre-determining functionality and building a monolithic application. This new approach has the potential to change how we design, deploy, and maintain mission system software.

*1 Computer Scientist, Intelligent Systems Division, MS 269-3, Ames Research Center Moffett Field CA 94035 USA

* 2 Flight Controller, Mission Operations Directorate, DS-15, NASA Johnson Space Center, Houston TX 77058 USA

MCT's approach to software will empower users and change the organizational dynamic between the IT organization and the users. By building software components that are both certified and end-user composable, IT providers can supply users with pre-certified pieces which the users can then put together as required to suit their needs.

II. The Users Perspective

MCT presents users with an environment of composable user objects. A *user object* is a piece of software that performs a function and may be assembled and combined with other user objects. This is not to be confused with a software object, which has meaning to programmers but not necessarily to end users.

The user environment includes all user objects accessible to a given user. Each project may implement policies to determine what a user will see and what they can manipulate. Figure 1 below shows an example of a user's environment.

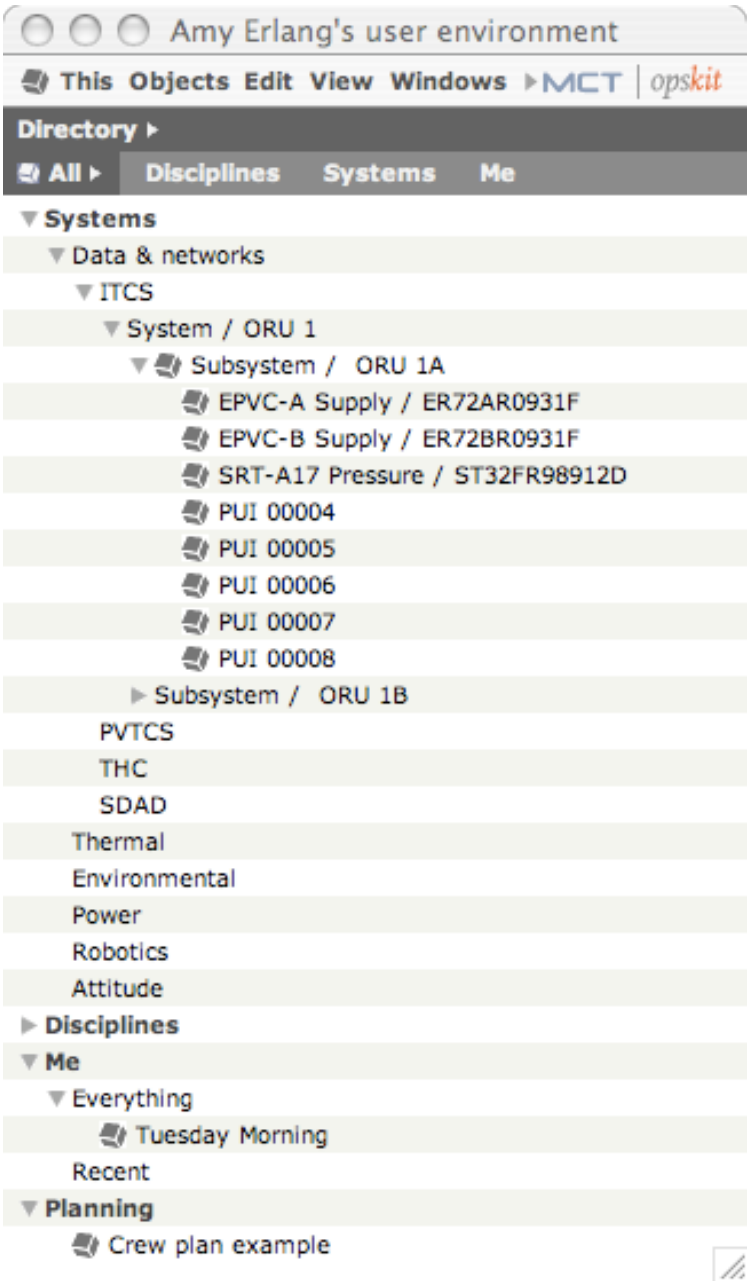

Figure 1 – User environment, sorted by classification of user objects

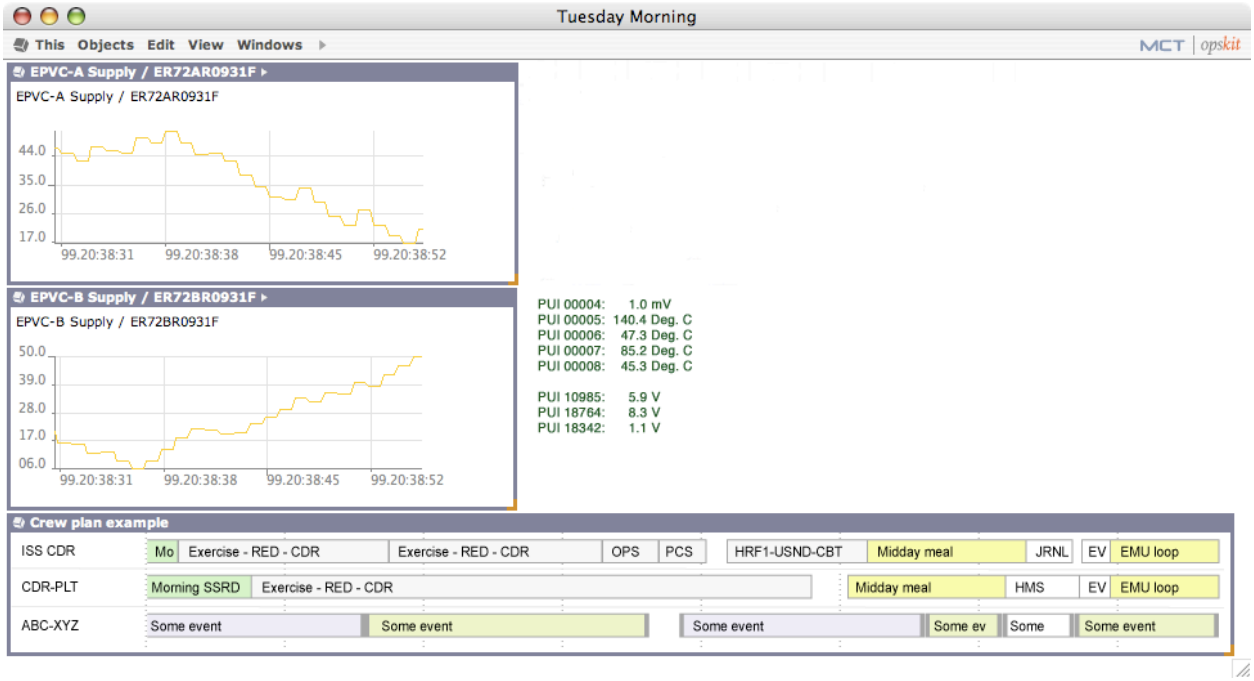

Figure 2 – Telemetry and monitoring display, with data plotted on the left and a corresponding alphanumeric view on the right

Figure 2 shows a telemetry and monitoring display. Each telemetry point in the display is a composable user object. Note the crew plan example in the display, showing how user objects from multiple domains can be combined into composite displays. Because user objects are live, their visible representation may be changed in place. For example, a telemetry object can be changed from an alphanumeric to a plot representation.

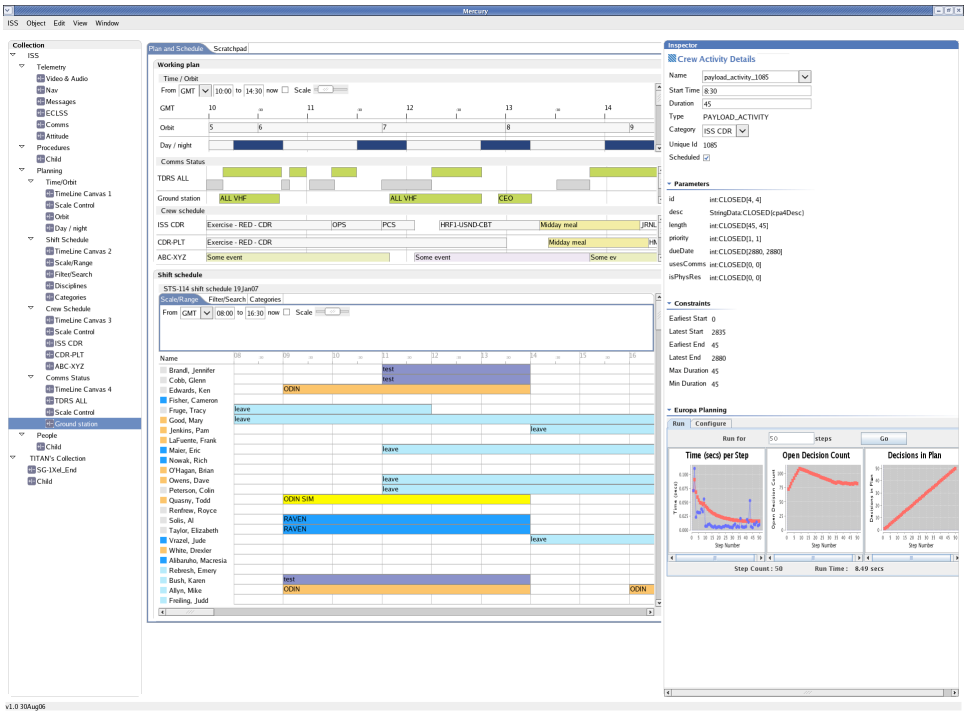

Figure 3 – A planning composition. Individual timeline user objects are combined with user objects to be displayed showing time properties. The user can drag a timeline from the left side of the screen, and populate it with crew activities and related events shown in graphical timeline representations.

III. The Architecture

From the user's perspective, MCT software is an environment of user objects that can be assembled into *compositions*. These compositions may or may not look like traditional software applications, depending on how they are assembled. Underneath what the user sees lies a set of core concepts that enables composable software.

A. Components

A software *component* is the fundamental unit of composition in MCT.

B. Roles

MCT components derive their characteristics from *roles*. A role is a set of characteristics or attributes. Roles can be thought of as similar to inheritance in object oriented programming. However, rather than inheriting characteristics through a hierarchy of classes, as in object oriented programming, with roles you simply have a list of roles in the system that any component may use and a component may have multiple roles. Some examples of roles are activity, or telemetry, view and model.

C. Model Components

A *model component* specifies data and logic. For example, a telemetry component displays a value corresponding to a measurement on board a vehicle. Its model component would define what the telemetry component is and how it processes the data stream. The visible manifestations

of a model component are shown to users as view components. This is similar to the idea of model in the model-view-controller (MVC) architecture.

D. View Components

A *view* is the visible representation of a model to a user. Because MCT components are "live" the user can change the visible representation in place without programming. See figure 2, which shows alphanumeric and plot representations of telemetry.

E. Composition Policies

Composition policies are rules stating what components may be combined and how. For example, an activity may be composed into a timeline component. An activity has a start and stop time which may be displayed in a timeline. A telemetry component embedded into a timeline could be displayed as a plot over time, or could show a point value at a specific time for a measurement that correlates to an event. Many types of compositions are possible.

F. Live Objects

MCT software compositions are assembled from user objects, which are "live" objects. Each "thing" which is composed is a working representation of an underlying model. While the user may select different representations, each representation is a different view of the same underlying thing. Figure 4 shows an example of a simple composition where a user drags a telemetry element onto a scratchpad, which is a blank slate for assembling components.

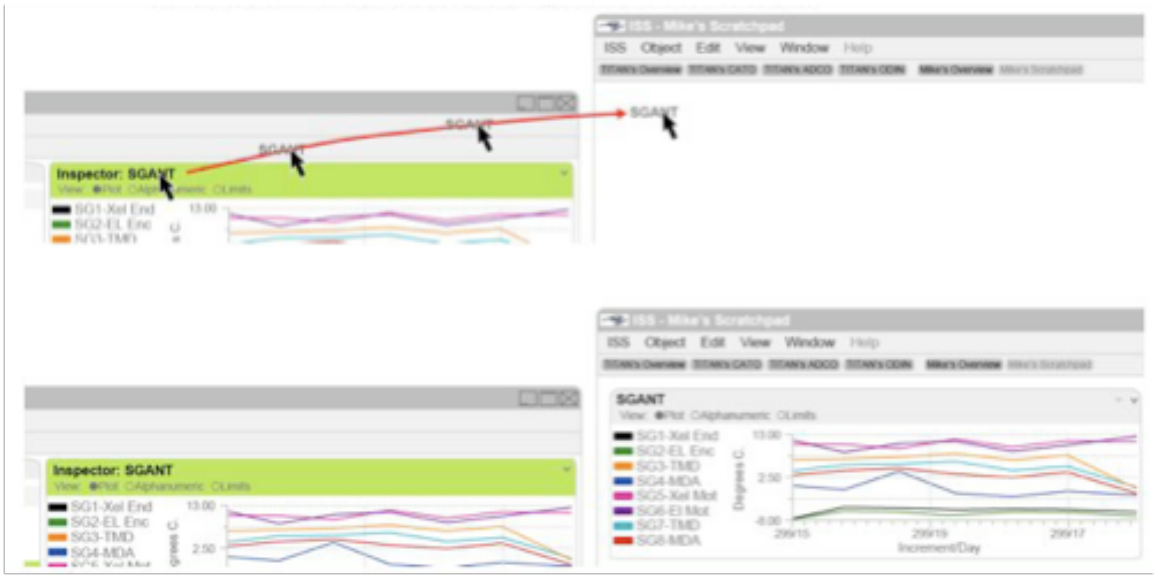

Figure 4 – Composition of parameters onto a "scratchpad"

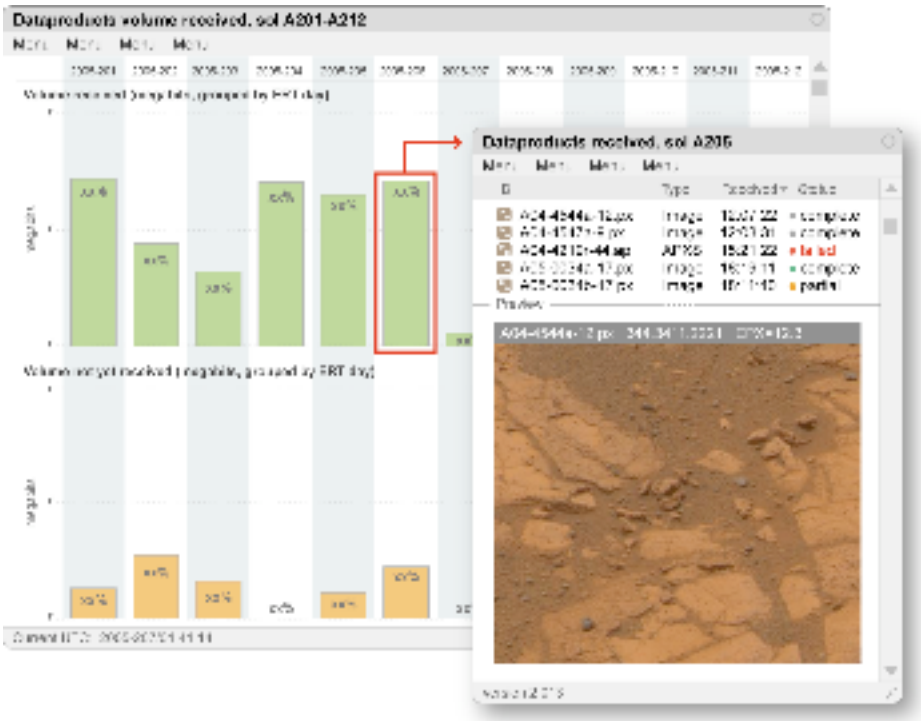

Figure 5 illustrates a data plot that is a composition of live objects. Clicking on the object shows the contained data.

IV. MCT Testing at Johnson Space Center

In June of 2007, MCT was tested in JSC's Operations Technology Facility (OTF), which is part of the Mission Control Center (MCC). The goals for the OTF engagement were:

- Integrate and run MCT in the OTF, shadowing the International Space Station (ISS)
- Develop, gather, and analyze measurements to evaluate the performance and usability of MCT, from a flight controller's perspective (within the scope of available FY07 time/resources).
- Define and analyze engineering metrics – performance, lines of code, potential cost savings (within the scope of available FY07 time/resources).
- Evaluate MCT's potential for improving the experience of flight controllers as they use mission control software, by having eleven flight controllers participate in two consecutive activities in the OTF.

Each person performed a set of ISS telemetry monitoring tasks. They then completed a questionnaire, designed using industry standards, about the capabilities and usability of the MCT user experience. Usability metrics evaluated were usefulness, usability, effectiveness, efficiency, satisfaction, and learnability.

Quantitative and subjective measures showed high potential benefits for the MCT approach. Users rated high satisfaction levels with learnability, ease of use, functionality and interaction design.

Engineering evaluations, and later evaluations using code analysis, showed that the MCT prototype used for evaluation in June of 2007 was well structured and maintainable. Performance was within MCC specifications. This was particularly important to verify, given the number of small components involved in MCT software. A memory leak was detected; this will be fixed in the transition from prototype to mission-level code.

IV. The Plan for Delivery

JSC's Mission Operations Directorate (MOD) is currently evaluating MCT as a promising candidate that may replace a large number of legacy MCC software tools. Assuming the formal selection of MCT, the following plan has been developed for delivery, deployment, and adoption of MCT.

1. Development of the core software infrastructure is in progress today to meet a Summer 2009 delivery that will be tested and certified for use in MCC. This delivery will support general telemetry monitoring functions including parameter plotting, event notification, limit indications, and related capabilities.

2. Once tested and certified for flight support, flight controllers will develop compositions for ISS operations. These compositions will be targeted to support specific MCC console positions. Following a test period, MCC will transition its ISS telemetry monitoring capabilities from separate legacy software tools to compositions built with MCT. This will allow JSC to gain significant operational experience with MCT before applying this new tool set to Constellation missions set to fly in 2013.

3. Although the initial 2009 capability delivery will be targeted for the MCC's Linux-based workstations, MCT is envisioned to support the office environment as well. A subsequent MCT software delivery, most likely in 2010, will extend the same MCT infrastructure and component set to the Windows-based office environment.

4. While this initial telemetry monitoring capability undergoes test and certification processes at JSC, work will continue at ARC to develop new components that will extend MCT capabilities beyond the limits of telemetry monitoring. The next set of deliveries will support user interfaces required for spacecraft commanding and procedure execution.

Acknowledgments

The MCT Team – Tom Dayton, Jack Hodges, Frank Robles, Irene Smith, Alan Tomotsugu, Sandy Johan, Sarah Hobart, Dennis Heher, Cori Schauer

Former MCT Team members – Harry Saddler, Steve Fonseca, Borek Vokach-Brodsky, Dave Curbow