SpaceOps 2010 Conference<br> <b><i>Delivering on the Dream</b></i><br> <i>Hosted by NASA Mars
25 - 30 April 2010, Huntsville, Alabama

AIAA 2010-2230

Development Of The Mission Operations Support Tool (MOST)

Trevor C. Sorensen[1], Eric J. Pilger[2], Mark S. Wood[3], Elizabeth D. Gregory[4], Miguel A. Nunes[5]
Hawaii Space Flight Laboratory, University of Hawaii, Honolulu, HI, 96822

The Hawaii Space Flight Laboratory (HSFL) was established at the University of Hawaii at Manoa in 2007 and is developing a launch vehicle and satellites. The second HSFL launch, scheduled for 2012, is STU-2, which includes a spacecraft being designed and built by the HSFL. Control of the HSFL missions will be done in the HSFL Mission Operations Center located on the University of Hawaii campus at Manoa. HSFL, in collaboration with NASA Ames Research Center and Santa Clara University, is developing a comprehensive open-architecture space mission operations system (COSMOS) to support this and future space missions. The major software tool of COSMOS, which is intended to provide real-time monitoring and control of spacecraft, is the Mission Operations Support Tool (MOST). This tool is based on the software tool LUNOPS which was designed for and used in support of science mission operations of the Clementine lunar mission in 1994. LUNOPS enabled the flight controllers to monitor the status of the spacecraft in accomplishing its science mission. MOST is building on this concept to allow not only monitoring of the spacecraft status, but also to provide a capability for issuing commands to the spacecraft and to be used in simulations, training and rehearsals, engineering data trending and archiving, and for anomaly resolution. The design goal for MOST is to create a single tool that can tie multiple data streams together with multiple end users. Toward this end, MOST is being designed to accept data inputs from multiple sources; while at the same time supporting multiple display configurations. On the data end, it can retrieve time stamped data records either from disk, or over established network protocols. These data can be archival, or real time; in the past, or in the future; real or simulated. MOST is capable of following data in real time, or tracking backwards and forwards in time. MOST supports one main overview screen, with summary data that is relevant to all users, plus multiple secondary screens designed for various support and subsystem tasks. The main display is always present, while the secondary screens can be displayed and dismissed at will. From the perspective of Mission Operations, this design allows each support specialty to hand tailor the display to their needs, while still maintaining access to all other information. From the perspective of monitoring and troubleshooting, the access to archival data allows studies of the interactions of different subsystems over time. MOST also provides a background monitoring mode that allows "lights-out" operation that will inform members of the operations team when an anomaly has been detected. Finally, the ability to work with simulated data allows the creation of virtual missions, for training, and support for forward looking for Mission Planning and testing.

I. Introduction

The Hawaii Space Flight Laboratory (HSFL) was established at the University of Hawaii at Manoa in 2007 for two primary purposes: (1) to educate students and help prepare them to enter the technical workforce, and (2) to help establish a viable space industry that will benefit the State of Hawaii. HSFL is currently developing a solid-propellant launch vehicle capable of placing small satellite (< 300 kg) into low Earth orbit (LEO). On the second

[1] Professor and Project Manager, AIAA Associate Fellow
[2] Flight Software Engineer
[3] Instrumentation Engineer
[4] Graduate Student, AIAA Student Member
[5] Graduate Student, AIAA Student Member

launch of this launch vehicle (designated launch STU-2) will be two satellites developed at the University of Hawaii: the "HawaiiSat" microsatellite from HSFL and the student-developed “Kumu A’o” CubeSat. Part of their mission requirements is to develop a means to operate the satellites during flight. Although several commercial and government-developed solutions were examined, none that were suitable fit within our highly-constrained budgets. A new solution was sought and found, which would be applicable not only to these two missions, but to other subsequent missions as well. This solution also provides a means for NASA Ames Research Center (ARC) to achieve their goal of developing a low-cost ground control and mission operations systems, which would contribute to the execution of multiple satellite payload missions currently being developed by NASA/ARC. This has led to a collaboration between HSFL and NASA/ARC for this project. Santa Clara University (SCU), which currently operates nanosatellites for NASA/ARC, is providing technical expertise for the project and actual satellite data for testing the operations software being developed by HSFL.

A. COSMOS

HSFL is developing a comprehensive open set of software tools with supporting hardware that is designed to primarily support the operations of one or more small spacecraft, but can also perform an important role in the design, development, and testing phases of spacecraft missions. This set of mission operations tools operate within an architecture named COSMOS (Comprehensive Open-architecture Space Mission Operations Support). COSMOS will be particularly suited for small operations teams with a very limited development and operations budget, such as universities. The COSMOS tools will initially be installed in two mission operation centers (MOCs), at HSFL and at NASA/ARC, in support of three satellites: University of Hawaii’s Kumu A’o CubeSat; a nanosatellite being developed by NASA/ARC; and HSFL’s HawaiiSat.

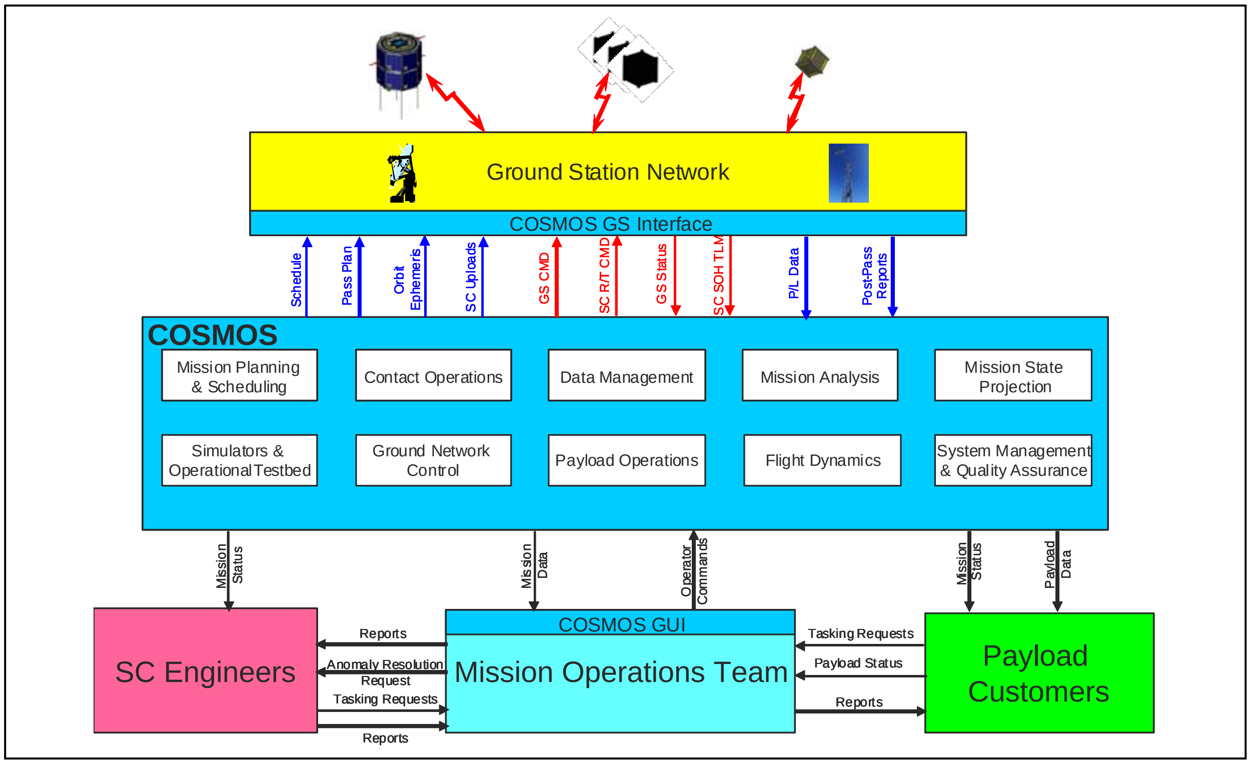


Figure 1. COSMOS Functional Architecture

The basic COSMOS functional architecture is shown in Figure 1. Within COSMOS the following major functions are performed/supported: mission planning and scheduling; contact operations; data management, mission analysis; mission state projection; simulations (including the operational testbed); ground network monitoring and control; payload operations, flight dynamics (including orbital and attitude); and support of system management and quality assurance.

The central pieces of this architecture are the visualization tools, support tools, and underlying programs that produce and manipulate the data needed by the rest of the tool sets. COSMOS combines both the software and unique hardware needed to support mission operations, including an operational test bed (OTB) and simulators.

The functional flow block diagram of COSMOS is shown in Figure 2. There are four major processes in mission operations that are supported by COSMOS:

1. *Mission Planning and Scheduling* which also includes command sequencing, the simulators, and the operational testbed (OTB);
2. *Contact Execution* which includes pre-contact operations, real-time contact operations, and post-contact operations both in the MOC and the ground network;
3. *Data Management* which includes transfer of all data throughout COSMOS and between COSMOS and external locations; data processing, such as engineering units conversion and Level 0 data processing; and data archiving;
4. *Mission Analysis* which includes support by the Mission Operations Team (MOT) to analyze and trend spacecraft and ground network state-of-health (SOH) data, orbital and trajectory data, and mission accomplishment data to help determine the mission success Measures of Effectiveness (MOEs). The results of the Mission Analysis process are fed back to the mission planners, spacecraft engineers (especially for resolving spacecraft anomalies), mission management, and customers.

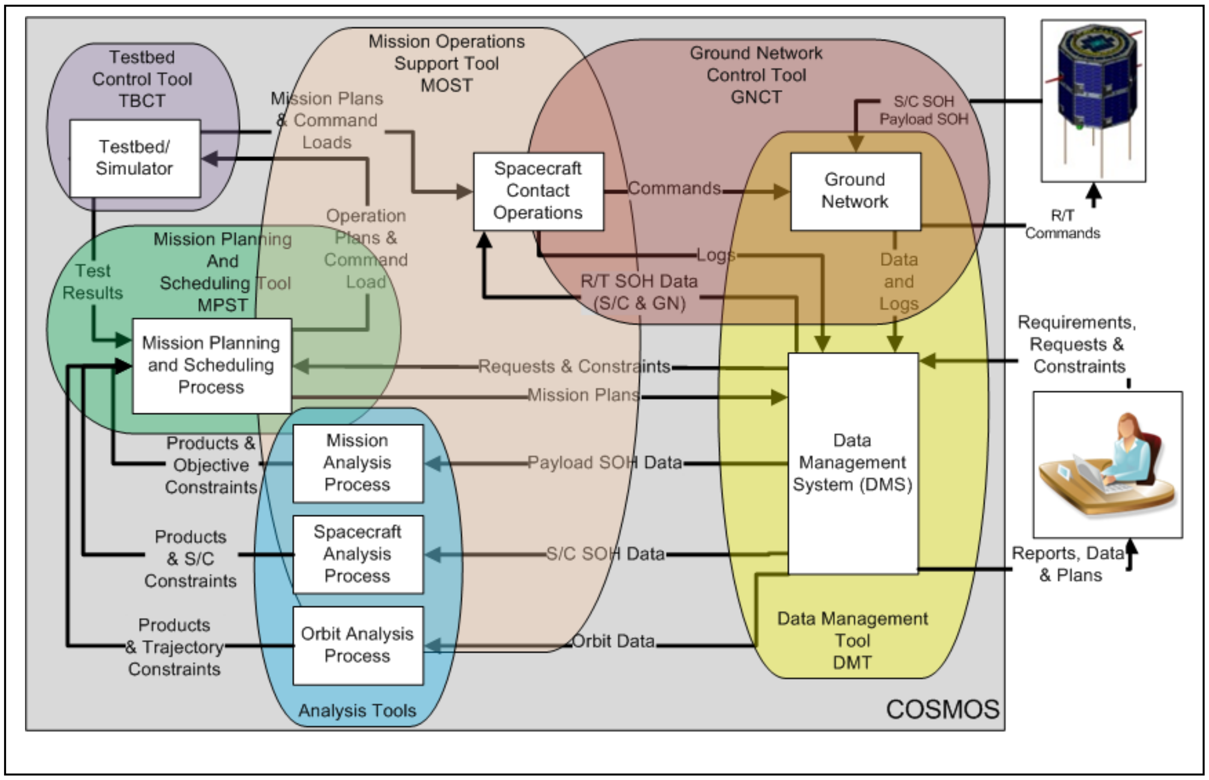

Testbed Control Tool TBCT
Testbed/ Simulator
Mission Operations Support Tool MOST
Mission Plans & Command Loads
Ground Network Control Tool GNCT
S/C SOH Payload SOH
Spacecraft Contact Operations
Commands
Ground Network
R/T Commands
Operation Plans & Command Load
Logs
Data and Logs
Test Results
Mission Planning And Scheduling Tool MPST
R/T SOH Data (S/C & GN)
Mission Planning and Scheduling Process
Requests & Constraints
Requirements, Requests & Constraints
Mission Plans
Products & Objective Constraints
Mission Analysis Process
Payload SOH Data
Data Management System (DMS)
Products & S/C Constraints
Spacecraft Analysis Process
S/C SOH Data
Reports, Data & Plans
Products & Trajectory Constraints
Orbit Analysis Process
Orbit Data
Data Management Tool DMT
Analysis Tools
COSMOS

Figure 2. COSMOS Functional Flow Block Diagram

Figure 2 also shows the primary tools that COSMOS provides for interfacing with the MOT to control these operations processes. The subject of this paper is the Mission Operations Support Tool (MOST) which covers the Spacecraft Contact Operations Function, part of the Mission Planning and Scheduling Function, and part of the Analysis Function. The rest of COSMOS provides the underlying processes and engines that move, generate, and process the data used by COSMOS and the MOT. It is anticipated that to handle multiple missions in a single MOC, each spacecraft will have its own session of the major COSMOS tools, either on the same or different consoles.

The basic philosophy behind the construction of this architecture, including MOST, is that its elements (tools and other programs) will be easy to port to a new location and to modify for operating with new spacecraft. This is enabled by being an “open architecture.” This approach means not only that the source code of its major elements and structure are available, but also that it is designed to accept external modules (which may not have source code available) as plug-ins through standard, well-defined interfaces in order to increase the overall capability of COSMOS for the desired application. However, it is recognized that there could be ITAR issues with COSMOS since it is designed to help control spacecraft. Therefore, we use a more limited definition of “open architecture” than the common one of having the source code in the public domain. We intend to provide COSMOS/MOST to only those entities (US government agencies, companies, or universities) which are allowable within ITAR restrictions. However, for those entities, the COSMOS/MOST source code will be available.

Spacecraft design, construction and subsequent mission support are all dependent on a number of unique protocols and technologies generally not found in Commercial-Off-The-Shelf (COTS) software. At the same time, although there is some significant overlap between the needs of all these activities, similar technologies are often used in markedly different ways, making tools incompatible at different phases, and leading to duplication of effort. As a result, tools for spacecraft design and mission support suffer from a number of flaws from the point of view of small missions developers. First among these is cost. While commercial packages are available, they come at a high buy-in and maintenance cost. A second flaw is that the existing tools are all of limited scope. Often two different tools that perform greatly similar functions will be required at different phases of the process. This serves not only to exacerbate the cost problem, but also leads to problems with integration, and with training. Finally, some areas, especially in mission support, are covered only by proprietary solutions, or not at all.

Although many universities operate their own small satellites, the operations systems are usually patched together using available COTS applications, such as MATLAB®, LabVIEW™, and MS® Excel, and are designed to be only sufficient to meet their immediate needs. MOST provides a solution that is being optimized from the beginning for mission operations and to add new and different types of spacecraft with minimum effort.

B. LUNOPS – The Parent of MOST

In 1993, Trevor Sorensen, the primary author of this paper, was the Lunar Mission Manager for the Department of Defense/NASA Clementine Mission being developed by the Naval Research Laboratory, the prime contractor. He was in direct charge of the Science Mission Operations and Planning (SMOP) group and was responsible for directing the Clementine Mission to accomplish the objectives of the Science Team while the spacecraft was in Lunar Orbit.[1] He realized that a tool was needed to help the MOT and the SMOP flight controllers in particular to efficiently monitor the progress of the spacecraft as it orbited the Moon and went through its various imaging sequences and other activities. The result was a program for the PC called LUNOPS, which he designed in August 1993, just five months before launch. Unfortunately at that time there were no programmers available to implement the design because Clementine was developed on a very lean budget.

Finally in November 1993 a young programmer was hired who came with experience from programming an advanced cruise missile. Unfortunately, after two months of work, this programmer was still “coming up to speed” and with a little over a week until launch of the Clementine spacecraft, had almost nothing to show for LUNOPS. Desperate times mean desperate measures, so Sorensen called on one of the scientists who was a member of his SMOP team, Dr. John Brandenburg, to take over the development of LUNOPS. Clementine launched on January 25, 1994. In about a week Brandenburg had LUNOPS Version 1 working, which was used in the MOC to show the position of Clementine around the Earth during its week of checkout before heading for the Moon. LUNOPS displayed count-down timers showing the time remaining until significant events like acquisition of signal (AOS), loss of signal (LOS) for the ground stations supporting the mission, onset of umbra (eclipse) and daylight, etc. This proved to be a very valuable tool during this phase of the mission, which was very stressful because Clementine was not designed for Earth orbit during which time its solar arrays were still undeployed.

During the approximately three weeks that it took for Clementine to travel from the Earth to the Moon, Brandenburg modified LUNOPS to support lunar operations as originally designed. Version 2, which had the basic functions working, was ready when Clementine entered lunar orbit. Improvements to LUNOPS were made during the course of the two and a half months lunar mission. The primary features of LUNOPS were also incorporated into the design of MOST. These features include:

- Orbital Position Display – shows the position of the spacecraft around the planet (Earth or Moon) that it is orbiting with planetary surface features displayed. Different views were available, which could be viewed simultaneously. The spacecraft position was determined from an ephemeris file that was read by LUNOPS.
- Spacecraft Attitude Display – in LUNOPS this was accomplished by drawing a vector from the symbol representing the spacecraft in orbit, the direction of which was the primary sensors direction (-z axis) and the length of which represented the magnitude of the vector in the orbital plane. When the sensors were pointing in their normal nadir attitude, the length of the vector would be a maximum. This vector was calculated from the attitude telemetry data received from Clementine. In MOST there is a separate attitude display that shows a depiction of the spacecraft with its orientation and several important vectors displayed.

- <u>Mission Events Display</u> – shows orbital events (latitude crossings, entry into and exit from umbra, etc.), mission events (AOS and LOS for various ground stations, imaging periods, etc.), and onboard commands, all with their associated execution times. The commands list was read in from a special file for LUNOPS that was generated from the Excel program that generated the orbit timeline. As an aid to the MOT, each event had a countdown (or countup) time and a current time bar across the display. It was possible to zoom in or out on this display as well as to move forward or backwards in time. The mission times and orbit number were also displayed.
- <u>Critical Parameters</u> – a few critical telemetry parameters were displayed.
- <u>Onboard Storage Display</u> – a time profile showing the status of the onboard storage used for storing images. This was important to monitor, especially on orbits when the spacecraft went through lunar occultation that precluded download of images thus reducing time available for image download. This particular feature was not operational by the end of the lunar mission.

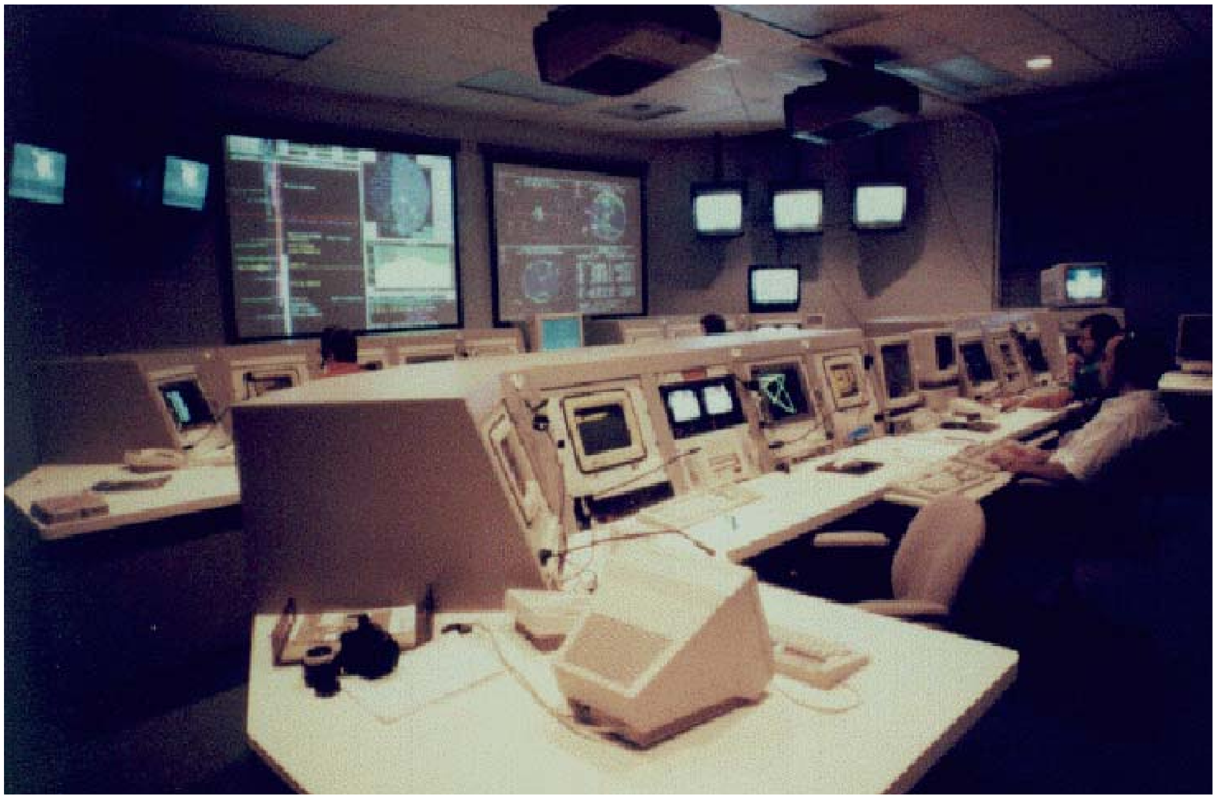


Figure 3. LUNOPS on Left Screen in the Clementine Mission Operations Center

II. MOST Description

A. MOST Architecture

The Mission Operations Support Tool (MOST) is the primary element of COSMOS and is the visualization and commanding tool designed specifically for supporting real-time operations. However, MOST can also be used for supporting the following major operations functions:

- spacecraft/payload monitor and control
- mission planning
- simulations and testing
- training and rehearsals
- trending and analysis
- anomaly resolution

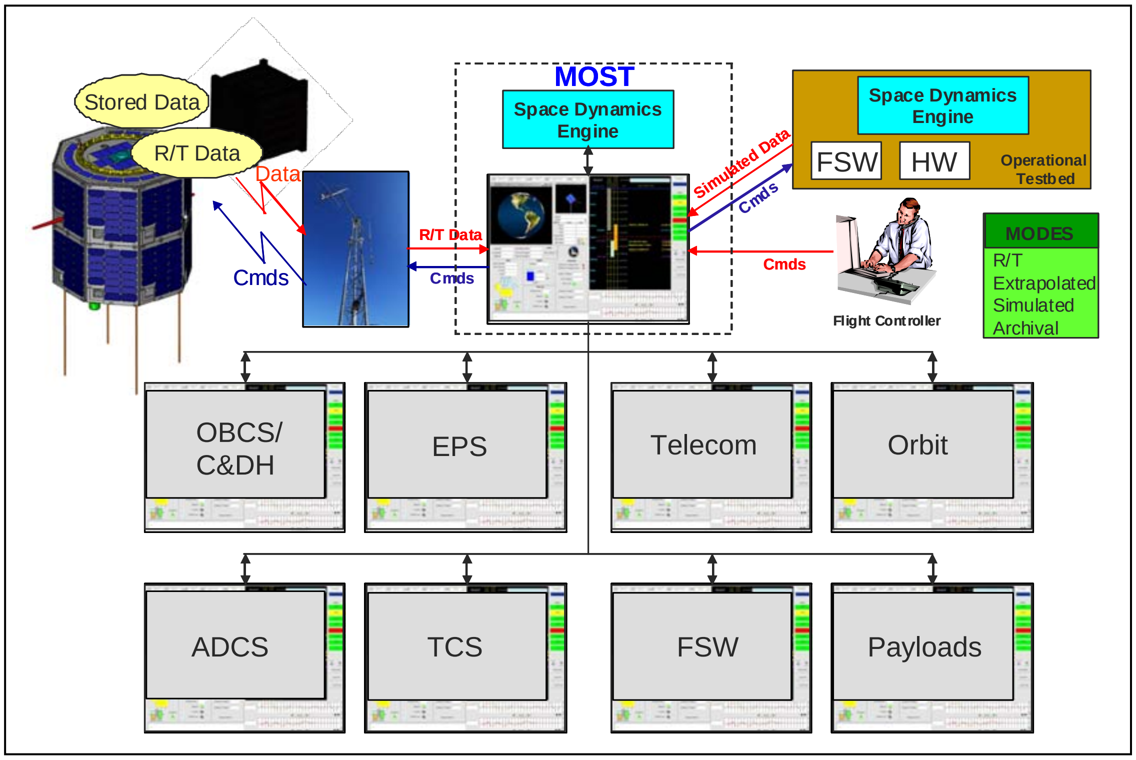


Figure 4. MOST Architecture

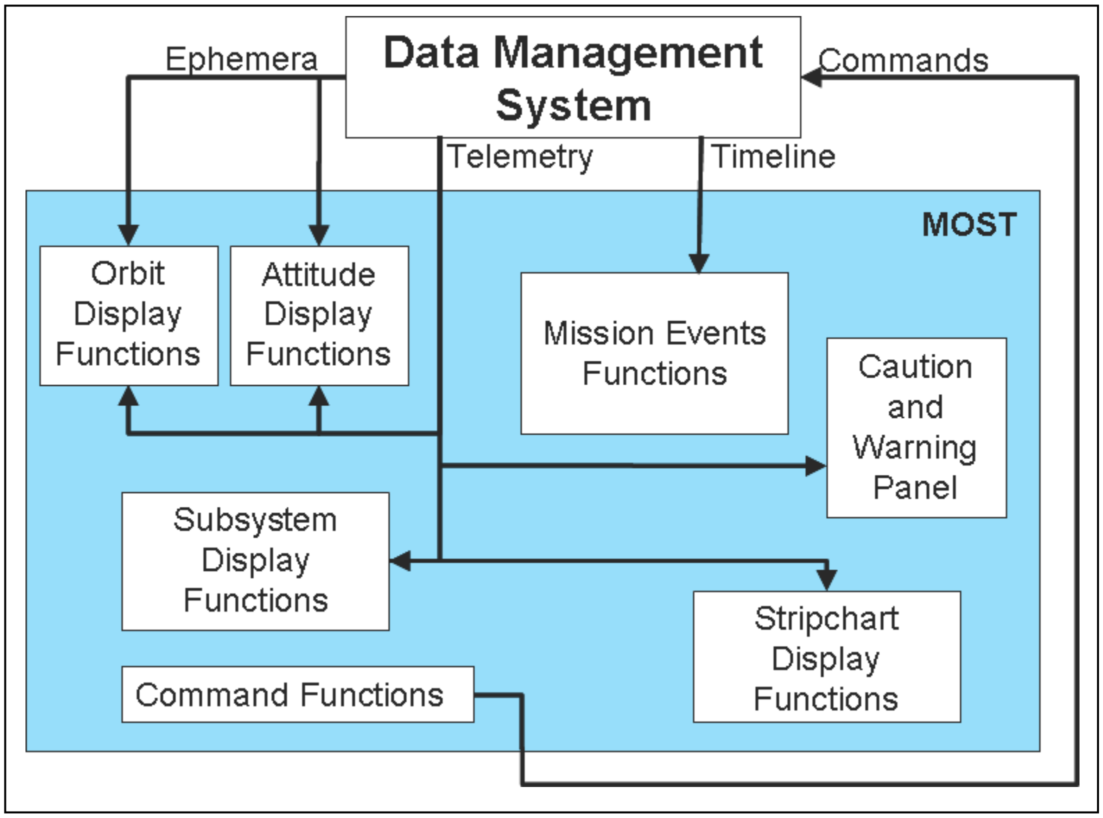


Figure 5. MOST Functional Block Diagram

MOST is based on the LUNOPS program, but incorporates more of the functions of real-time monitoring and control of the spacecraft bus including the generation and sending of real-time commands, which LUNOPS did not.

The architecture of MOST is shown in Figure 4. MOST has several different modes of operation, as shown on the right side of the figure. For (near) real-time operations it interfaces with the spacecraft through the Ground Network. The MOST Functional Block Diagram is shown in Figure 5. The real-time telemetry is streamed directly into MOST through the Data Management System (DMS). Stored telemetry data also comes from the spacecraft through the Ground Network and DMS. Archival data are telemetry data that have been stored in the mission archive of the DMS and can also be displayed by MOST. Simulated data come from the simulators or OTB. If data extrapolated from actual telemetry data are desired, then the simulators can be used to produce these data as well. MOST has access to the Space Dynamics Engine, which calculates the position and attitude of the spacecraft based on flight dynamics and environmental models, for simulated or extrapolated data when simulators are unavailable or unnecessary. MOST defaults to a mission overview screen (Figure 6), but dedicated windows for all subsystems or elements can be displayed as well.

The MOST mission overview screen shown in Figure 6 has five basic functions:

1) Show a timeline with past and future events, including uploaded commands.
2) Display subsystem and payload status.
3) Provide a visual/graphical display of the spacecraft orbit and attitude.
4) Detect anomalies and display warnings pertinent to spacecraft conditions.
5) Send real-time commands to the spacecraft.

The timeline chart (Mission Events Display) shows past and future events, both orbit related and command related. On the left side of this display are the orbit events including passing into and out of eclipse (umbra) as well as ground contact events (AOS or LOS). Next are some vertical bars showing the time period covered by these orbital events. These event bars are generated by MOST. One or more payload or spacecraft event bars can be added on the right side. Next comes the time scale in UTC and then the list of spacecraft commands (usually from the onboard command sequence). On the far right are the countdown (or countup) times for the various events, which are calculated by MOST. The current time is shown by a red horizontal bar. It is possible to zoom in or out of this display to set the desired resolution, and to move forward or backwards in time. At the top of the Mission Events Display (MED) is a window that displays the current spacecraft mode (state) as defined by the flight software.

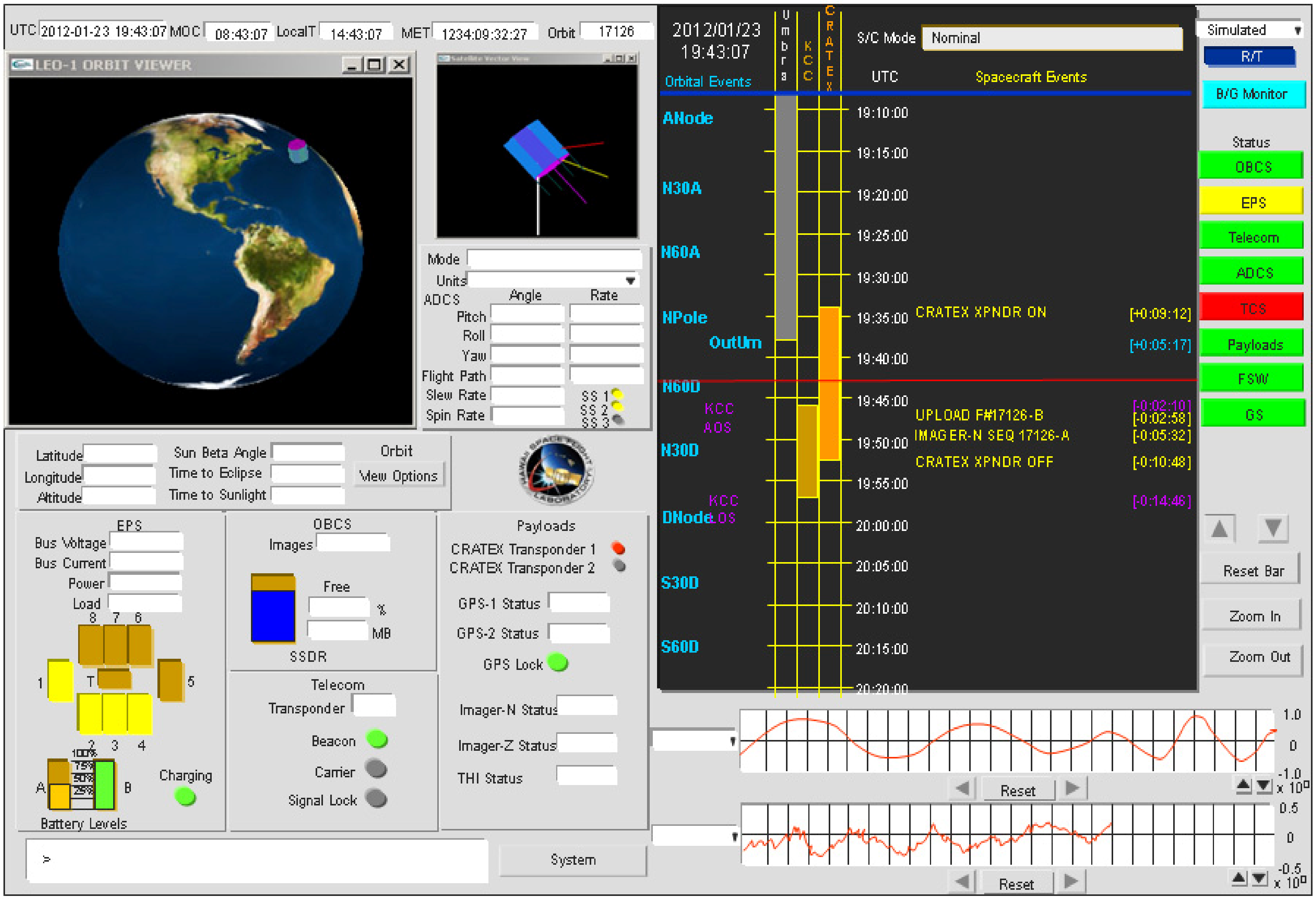
The diagrams and text boxes (called panels) on the lower left quadrant display the spacecraft subsystem and payload status (from telemetry data received from the spacecraft) and are defined by the user for the subsystems and data to be displayed during the MOST setup and configuration. If the user (Flight Controller) clicks on one of these subsystem panels then that subsystem window is displayed as an overlay on the main display. Figure 7 shows the subsystem window for the HawaiiSat Thermal Control Subsystem (TCS) from the MOST prototype program.

On the lower right are two strip chart displays which allow the user to select any two telemetry parameters to be displayed as a time history strip chart. The time scale and parameter range of the strip chart are user-definable. The user can also move backwards in time to view additional history.

The two 3-D windows on the upper left of the mission overview display show the satellite position with respect to Earth (Orbit Display) and its attitude (Attitude Display) with the values of the essential parameters. Both of these displays can be modified by the user to show different perspectives and to select other viewing options.

A Caution and Warnings (C&W) Panel on the far right contains colored push buttons which are indicators for various subsystems. These status lights are based on the limits testing that MOST does on input telemetry. If all telemetry parameters are within nominal operating range, then the status lights are green. If a parameter is detected in the telemetry that has just exceeded the caution (yellow) or warning (red) limits set for it, then a warning window is displayed which shows the parameter, its value, and the limits value. The status lights then stay this same color until the parameter value passes another threshold value. When a C&W button is pressed, a window for that subsystem is opened which contains comprehensive information about that subsystem.

On the bottom left section of the MOST main display is a window with a prompt. This is where the user can enter commands that are to be sent in real-time to the spacecraft during a contact. This would typically be used in an emergency, like when the scheduled downlink of data needs to be stopped because of a system problem. Any command entered here undergoes syntax and validity check as well as confirmation before being sent. Sometimes it is necessary to send a series of commands or build a command macro to be sent. To do this the user clicks on the command panel and a Commands window will be displayed. This has the full dictionary of spacecraft commands and also provides the ability to put several command lines together into a macro, which can then be sent to the spacecraft via the ground network as a group.



UTC 2012-01-23 19:43:07 MOC 08:43:07 LocalT 14:43:07 MET 1234:09:32:27 Orbit 17126

2012/01/23 19:43:07

S/C Mode Nominal

```
19:10:00
19:15:00
19:20:00
19:25:00
19:30:00
19:35:00  CRATEX XPNDR ON          [+0:09:12]
                                   [+0:05:17]
19:40:00
19:45:00  UPLOAD F#17126-B         [-0:02:10]
                                   [-0:02:58]
19:50:00  IMAGER-N SEQ 17126-A     [-0:05:32]
          CRATEX XPNDR OFF         [-0:10:48]
19:55:00                           [-0:14:46]
20:00:00
20:05:00
20:10:00
20:15:00
20:20:00
```

Figure 6. Design Layout for MOST Main Overview

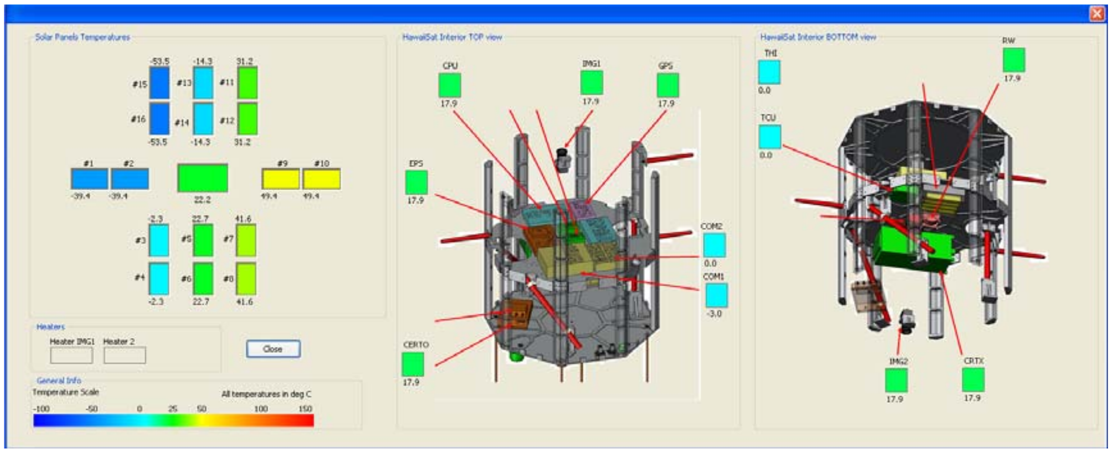


Figure 7. Prototype MOST Displays for the HawaiiSat TCS

B. Concept of Operation

Although the primary purpose of MOST is to be a monitor and control tool for spacecraft real-time operations, the versatility of MOST means that it can be used to support other areas of mission operations as well. These include mission planning; simulations; training; rehearsals; spacecraft mission and SOH trending and analysis; and anomaly resolution. To support these other functions, different modes of operation can be specified within MOST, which essentially identify the source and type of data to be used by MOST. The modes are:

- *Real-Time (R/T)* – which is actually a "Near Real-Time" mode during which MOST displays streaming data from a spacecraft during a contact with a ground station and passed through the ground network to the MOC. These data are the real-time data and not the stored data being downlinked from the spacecraft. Once contact with the spacecraft is lost, the last values received continue to be displayed, but MOST dulls the visual output to indicate that the values are not currently being updated. This mode is used to support real-time operations.
- *Extrapolated* – MOST has the ability to take the latest real-time data and extrapolate it into the future so that the user can see probable conditions of the spacecraft at some future time. MOST uses either simple extrapolation techniques for independent variables such as time, or uses the models in the spacecraft simulator to calculate values. Not all variables may be available in *Extrapolated* mode – it depends on the implementation and how comprehensive the simulation models are. This mode can be used to support mission planning, real-time operations or for simulations, training, analysis, etc.
- *Simulated* – This mode indicates that the data being displayed by MOST are simulated data being received from the OTB or simulator. In all other ways (depending on the MOST settings) MOST behaves as if these are real-time data. This mode is used when testing command scripts as part of the mission planning process, and is also used for training, rehearsals, and looking at hypothetical cases during anomaly resolution.
- *Archival* – In this mode MOST reads in stored spacecraft data rather than real-time data. MOST allows a user to scan back through archived data, and present the data at a speed that the user chooses. This mode is most important for supporting trending and analysis of SOH or payload data, anomaly resolution, and may also be helpful in certain circumstances for mission planning.

MOST updates the data displayed by providing its own internal clock that can be set to real time, or any desired time. The clock speed can be sped up, slowed down, or stopped. In *R/T* mode, the MOST clock is set to real time. In *Simulated* mode, *Archival* mode, or *Extrapolated* mode, the MOST clock is set to a time specified by the user. The speed at which to display the data can also be selected. MOST then shows (or "plays") the images and data for the user to see.

There is an additional special mode that is not related to the display of data – the *Background (B/G) Monitoring* mode. This mode enables MOST to monitor R/T streaming data while it is being used in any mode. Thus if the user is looking at archival data during a long contact and a spacecraft parameter suddenly goes out of limit, then MOST will detect this and initiate a sequence of events including changing the subsystem status light and displaying a pop-up window that contains details of the detected anomaly. This mode is also essential for "lights-out" operations for when the MOC is unattended. MOST has a Notification window through which one or more recipients can be specified to receive a text message initiated by MOST which details the anomaly, including time, which parameter, current value, and threshold value.

III. Program Design and Implementation

A. Functional Design

One of the primary aims of the MOST development is to enable MOTs to have all the necessary information for a single Flight Controller to conduct monitoring and control operations of a spacecraft with a single tool on a single console. This requires all the information required to be presented in a clear and logical fashion tailored to the needs of the mission. The overall status of the spacecraft is available, but with a single click, the Flight Controller can access detailed information about the spacecraft and mission. This enables a small operations team to control relatively complex spacecraft with a minimum of labor and cost.

MOST uses a configuration file to tailor it to the needs of a specific spacecraft. The configuration file is read in by MOST at startup, and gives MOST specific parameters that it uses to configure itself for a given spacecraft. These parameters tell MOST which information diagrams, text boxes, timelines, graphics windows, and subsystem dialog boxes will be present, and where they will be placed on the MOST graphical user interface (GUI).

Once MOST configures itself, it runs continuously, performing the following functions at various rates:

1. Load spacecraft data from requested time period into viewing memory window.
2. Load any new spacecraft data into real time memory window.
3. Set pointer to appropriate time in either viewing or real time window, based on current time flow.
4. Perform any required calculations and display for current mode.
5. Perform any requested comparisons and deliveries of alerts for background monitoring.

MOST gets its data for requested time periods in one of two ways. Past data are read from the data archive, which is kept in a location specified by the MOST configuration file. Future data can either be read from an ephemeris, stored in the same location as the archive, or generated internally. MOST acquires its new data when the spacecraft is in contact with a ground station, at which point it will read live telemetry data over a network connection.

MOST sets its time pointer from a main execution loop. The loop is executed once for every tick of the MOST clock. When MOST runs in *R/T* mode, the MOST clock is the same as real time. In the other modes, the MOST clock is set to a time and speed specified by the user. MOST then displays data that corresponds to this clock. The main loop also calls functions to determine if subsystem parameters are at an alert or emergency condition. Each time through the loop, if it is determined that any values are out of range (as defined by the configuration file), the lit buttons on the Caution and Warning Panel will change colors and in the event of an emergency condition will flash red. Any requested comparisons and deliveries of alerts will also be run at this time.

A major component of MOST is the Mission Events Display (MED), including the Command Timeline. The design of the MED uses a bitmapped space with a linear timescale conceptually mapped into it. The time scale runs vertically (from top to bottom), and is divided into several columns for the different types of data. Events, times, and countdown timers are printed onto this “map” in the proper columns (depending on the type of data) at the proper “time” location.

A slide bar and zoom buttons at the right side of the MED allow the user to change the time space they are viewing. When the program is started, MOST reads in the last eight hours worth of data, and uses this to allow the MED to show a span of eight hours (ending at the present time). MOST continues to update and show the last eight hours of data on the Command Timeline. If the user decides to zoom out, this causes the MED to show more than eight hours of data, and MOST reads the additional data from the archive files. If the user slides the slide bar into the past (away from “present” time), MOST reads in the archival data as needed. In this case, MOST changes its internal clock time to the time specified by the user, and runs at the user specified speed. The user has the option of speeding up, slowing down, or stopping the clock.

B. Design Implementation

To facilitate usage on multiple platforms, MOST is being written in Qt as a limited open source project. To comply with ITAR regulations, MOST will be licensed with a Limited GNU Public License.

Qt is a cross-platform application and User Interface (UI) framework that is used to develop GUI interfaces as well as non-GUI programs as servers and consoles. Qt uses standard C++ class libraries. This allows portability

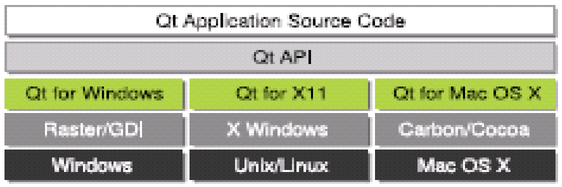


Figure 8. Software architecture using Qt (ref. 2)

between different operating systems, namely Windows, Mac Os X and Linux (Figure 8). The modular Qt C++ class library provides an extensive set of system applications that make possible the functionality needed to build cross-platform and advanced applications like MOST.

The development of the software is made using Qt Creator, a cross-platform C++ integrated development environment that is part of the Qt Software Development Kit (SDK). Using Qt Creator allows standard development procedures for the different operative platforms and allows for a clear setup of the UI environment. It uses the C++ compiler from the GNU Compiler Collection on Linux and MinGW on Windows that has been successfully used in the software industry for many years.

To further support MOST in its goal of being a flexible visualization tool for multiple spacecraft, it has to be designed on a framework that is limited enough to be easily defined, while still being complete enough to cover the desired characteristics of the spacecraft being represented. It is therefore very important to define a basic set of building blocks, around which the Interface can be built. These building blocks need to be complete enough to represent all elements of the spacecraft in which we are interested. At the same time they must remain simple enough to retain a direct relation to the overlying GUI. Finally, the whole has to be designed with flexibility in mind for future expansion.

To meet this goal, the software team is working on a suite of four design components to support the lowest level of the software structure. These components interact with each other to provide a solid framework from which the GUI derives its structure.

The first component is a Structural Elements Definition (SED). The purpose of this definition is to define all elements of the spacecraft that we might want to represent to the GUI. This is not an attempt to represent every aspect of the spacecraft. Instead, we will list the common aspects of all the spacecraft that we would like to represent in MOST. The SED includes, at a minimum, key structures, such as panels, and components, along with related values of shape and position, mass, and power. It is complete enough to provide a template of all the pieces required to roughly model the behavior of a given spacecraft, and to represent its functioning in MOST.

An example SED, as derived from a standard small satellite, might consist of the following:

- Structural Panels
  - Mass
  - Corners (in body frame)
  - Thermal constants
  - Temperature
  - Interior/Exterior
- Structural Boxes
  - Mass
  - Corners (in body frame)
  - Thermal constants
  - Temperature
- Solar Panels
  - Mass
  - Corners
  - Thermal constants
  - Temperature
  - Electrical characteristics
  - Structural Panel
- Electronic Devices
  - Mass
  - Location
  - Electrical characteristics
  - State
  - Temperature
  - Structural Box

The full SED for MOST will have additional elements as derived from the complete cross section of spacecraft supported. With a well-defined SED, it is possible to capture the entire state of the spacecraft being observed. However, for it to be displayed in MOST, it has to be tied to code. This is achieved through the additional components described below.

The second component, a well-defined Data Name Space (DNS), is what allows the SED, as well as all other aspects of the spacecraft state, to be mapped into software. This step involves both the creation of a variable to represent each element, and the placement of each variable in a well-ordered structure. It also involves the careful naming of each variable, for both internal representation, and listing in external data streams and files. While the internal naming scheme can take advantage of hierarchical relationships to allow for reuse of names (such as x, y, z), the external scheme requires unique names for ease of representation in a simplified JavaScript Object Notation (JSON). It is important that both naming schemes be expandable, for representation of multiple elements, and that they map to each other.

Part of an example naming scheme, is shown in Table 1. This demonstrates the mapping between the unique DNS names and the internal software representation of some spacecraft state variables. The first half shows how one of many structural panels would be represented, while the second half shows the geodetic location of the spacecraft.

Table 1. Example Spacecraft Naming Scheme

External (DNS)	Internal Code Structure
Spacecraft structural panel	
panel_01_corner_01_x	panel #1 of n: cornerl #1 of m : x value
panel_01_corner_01_y	panel #1 of n : cornerl #1 of m : y value
panel_01_corner_01_z	panel #1 of n : cornerl #1 of m : z value
panel_01_corner_02_x	panel #1 of n : cornerl #2 of m : x value
panel_01_corner_02_y	panel #1 of n : cornerl #2 of m : y value
panel_01_corner_02_z	panel #1 of n : cornerl #2 of m : z value
panel_01_thermal_c	panel #1 of n : thermal conductivity
panel_01_thermal_s	panel #1 of n : specific heat
panel_01_temp	panel #1 of n : temperature
panel_01_position	panel #1 of n : external/internal
.	
Spacecraft geodetic location	
geod_pos_x	geodetic location : position : x value
geod_pos_y	geodetic location : position : y value
geod_pos_z	geodetic location : position : z value
geod_vel_x	geodetic location : velocity : x value
geod_vel_y	geodetic location : velocity : y value
geod_vel_z	geodetic location : velocity : z value
geod_acc_x	geodetic location : acceleration : x value
geod_acc_y	geodetic location : acceleration : y value
geod_acc_z	geodetic location : acceleration : z value

The two components listed above are sufficient to provide a static representation of the state of the spacecraft. To really bring the display alive, operations are needed. These are provided by the third component, functional libraries. To enhance the display of data from the spacecraft, software provides a library of function calls spanning a range of complexity. The most basic supply conversions from one representation to another, and comparisons to threshold levels. The most complex propagate orbital position and attitude. These libraries will be developed as part of the larger COSMOS effort.

Finally, in order to meet the MOST goal of easy representation of multiple spacecraft, there needs to be some way to tie the various components together without having to recompile the software. The fourth design component is therefore a configuration scheme that allows the DNS and SED to be mapped to specific elements of a specific spacecraft. The configuration scheme includes, at a minimum, a spacecraft configuration file, a user interface configuration file, and various standard images for use in the interface. The spacecraft configuration file is an ASCII file that contains, at a minimum:

- A one to one connection between parts of the spacecraft and External Names as defined in the DNS
- The location of each ground station expected to be in use
- The names of any special image files that will be loaded in to the interface
- The name of the user interface file

The user interface file is a standard Qt interface file. This is either a stock file, created when the original spacecraft configuration was done, or the output of the MOST Editor. Either way, it dictates the arrangement of controls and displays, and the linking to specific images and values, for the specified spacecraft. The various images are for display of stock items such as solar panels, thermal sensors, and devices.

These components, if well thought out, allow a single version of the MOST software to flexibly represent a wide range of behavior in multiple spacecraft. Furthermore, once the configuration for a given spacecraft is complete, it is straightforward to either switch the interface from one spacecraft to another, or to start a separate session of MOST. Finally, the configuration for a spacecraft is well within the abilities of anyone with reasonable computer knowledge. A simple configuration requires nothing more than a text or MOST/Qt editor and a photo processing program.

IV. Modifying MOST for New Spacecraft

A. Modification Philosophy

The goal of MOST development is to create a framework that is easily adapted and customized for a wide variety of spacecraft applications. MOST, in its initial development, is being built for HawaiiSat, but wherever possible, customizable options are integrated to allow the structure to be changed to accommodate different payloads and spacecraft containing more or fewer subsystems.

Unlike with many other programs MOST does not dictate to the user what they can see or do but allows the user to do what needs to be done for their unique spacecraft. Part B of this section discusses in detail the MOST Editor and the capability for MOST to be modified for each unique spacecraft and to adapt to changing mission needs.

During the development of the spacecraft all the subsystem and payload controllers provide input for what they would like to see in their subsystem window and on in their panels on the main display. The MOST implementer takes the input from all the controllers and customizes the standard MOST interface for use with the unique mission, spacecraft, or organization. MOST can also be easily modified during mission operations to accommodate unforeseen issues and operations. For example, if during operations there is a malfunction and a certain aspect of the spacecraft requires close monitoring, these data can be moved to the main display or tracked with a strip chart.

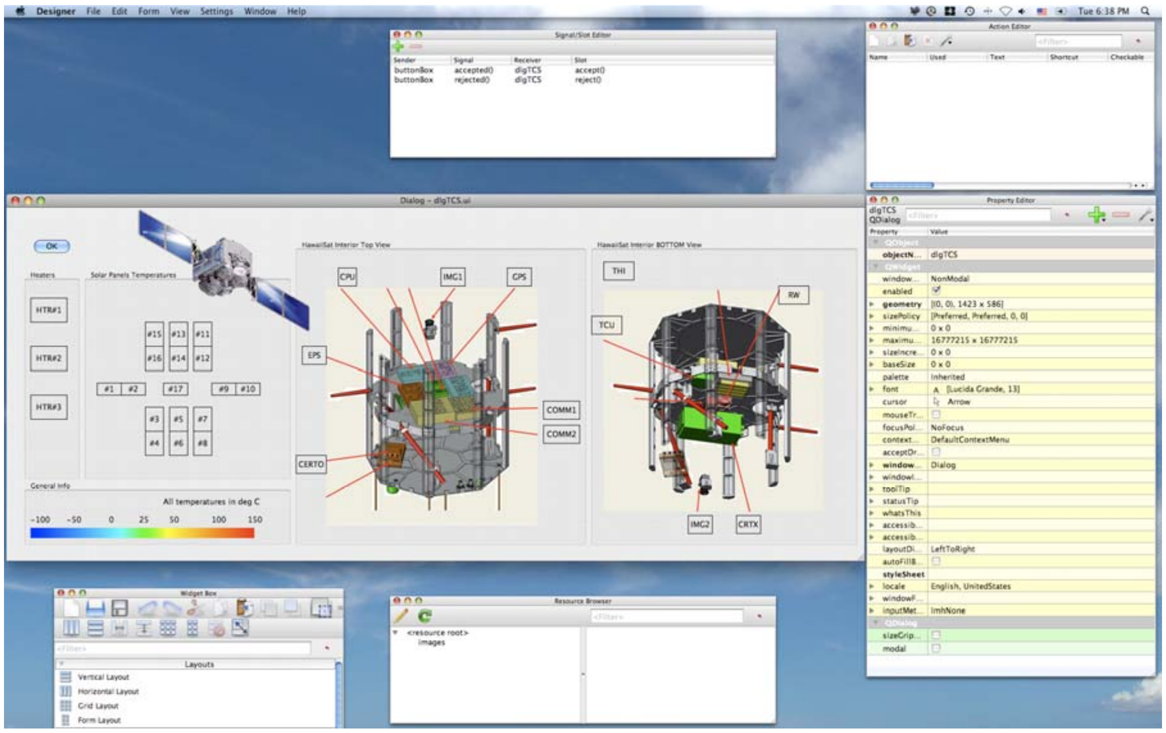
MOST is a versatile tool that allows users to accomplish all of their mission operations goals. The modification process for MOST will continually be developed to create an ever more intuitive environment for modification with the least amount of interruption to operations. Modification to the standard interface will be simple and applicable at any point on the design and operation process as required.

B. COSMOS/MOST Editor

The capability to modify MOST interfaces will be made possible with the MOST Editor. This is a modular software tool that is capable of changing its own interface and internal configuration to operate the various necessary missions. Any user will be capable to create or adapting the standard configuration file and user interface files provided with MOST to its own spacecraft mission specifications.

The capability of dynamically changing the user interface form dialog is also specified as run-time form processing, because the forms are processed at run-time and produce the dynamically-generated interfaces based on the input provided. This dynamic process is made possible through the embedded classes (like QtUiTools) and modules. These enable the form to be processed at run-time by changing the user interface file (.ui) and then load it into the run-time environment. The capability of creating a user interface is also available using the Qt Designer platform that is the default user interface editor for Qt (downloadable for free at http://qt.nokia.com/downloads). Both tools (MOST Editor and Qt Designer) make it easy to develop any user interface with the Qt's signals and slots mechanism for type-safe communication between the graphic objects.

The MOST Editor is structured using the Qt environment and it contains a series of default templates, as a visual interface library, for different spacecraft missions. It also contains the corresponding subsystems allowing the user to easily put together the necessary graphic interfaces and data variables for its mission. The process of creating a new mission environment will be standardized and guided with specific procedures. Tutorials may be created and added to the database of previous created mission environments as open source and free to use. Typical environments that will be part of the interface library are: Main Dialog (showing the main subsystems and a summarized description of their status); Electrical Power Subsystem; Attitude and Determination Control Subsystem; Communications Subsystem; Thermal Control Subsystem; etc. The editor also allows the user to create a new subsystem (normally a payload) making it scalable and very flexible for any particular needs.



Sender	Signal	Receiver	Slot
buttonBox	accepted()	dlgTCS	accept()
buttonBox	rejected()	dlgTCS	reject()

dlgTCS
QDialog

Property	Value
QObject	
objectN...	dlgTCS
QWidget	
window...	NonModal
enabled	✓
geometry	[(0, 0), 1423 x 586]
sizePolicy	[Preferred, Preferred, 0, 0]
minimu...	0 x 0
maximu...	16777215 x 16777215
sizeIncre...	0 x 0
baseSize	0 x 0
palette	Inherited
font	A [Lucida Grande, 13]
cursor	Arrow
mouseTr...	☐
focusPol...	NoFocus
context...	DefaultContextMenu
acceptDr...	☐
window...	Dialog
windowI...	
toolTip	
statusTip	
whatsThis	
accessib...	
accessib...	
layoutDi...	LeftToRight
autoFillB...	☐
styleSheet	
locale	English, UnitedStates
windowF...	
inputMet...	ImhNone
QDialog	
sizeGrip...	☐
modal	☐

Figure 10. MOST Editor Prototype Using Qt Editor

Figure 10 shows one example of the construction of the user interface for the Thermal Control Subsystem for the HawaiiSat mission. This user interface is stored in an xml standard file format that can be read in run-time when the MOST software is operating. The satellite figures can be easily changed as well as the temperature boxes. It is also possible to add more sensor displays and/or figures. This shows how the user interface files can be easily edited or created.

After being validated, the MOST Editor will be expanded to become the COSMOS Editor using the same paradigm as specified in this section allowing not only the creating of simple graphic and data interfaces but a more complex structure that is necessary for a more complete set of tools when surveying a spacecraft in the mission operation scheme.

V. MOST Development Plan

A. MOST Prototype Development

MOST is currently a work in progress, and it will remain so for some time, since the end goal is a product that will be highly configurable, and quite feature laden. Its development process needs to account for this, providing for quick initial results, while allowing plenty of room for expansion later. In addition, its primary purpose is as a visualization tool for large sets of data from orbiting spacecraft. The quick initial results need to include some form of these data. Finally, the design team itself needed to build on their existing capabilities, advancing hand in hand with MOST.

To accommodate these various requirements, the design process divided MOST development into two major sections: display, and data. Both sections have then gone through an iterative process of prototyping, followed by implementation. As more features have been implemented, our prototyping environment has changed to keep pace.

Microsoft® Visual Studio was used as the first prototype environment for the display section of the software, basing the look and feel on the LUNOPS software (as discussed previously), and suggestions from the design team. As there was no existing set of spacecraft data to visualize, a Space Dynamics Engine was developed, with an underlying orbital propagator, to provide orbital and attitude data. This was compiled directly in to the original prototype so that the team could get a feel for performance under real conditions.

Once the initial prototype was working, it was decided that changes needed to be made on both the display and data sides. On the display side, the team decided that a more flexible and open solution was needed to meet the open-architecture goals of MOST. It was therefore decided to investigate migration to the Qt development system. At the same time, since it was clear that MOST was being developed to process external information, work was started on an external version of the Space Dynamics Engine. A first prototype of this was completed, along with a rudimentary DNS, and MOST was transitioned to reading files generated by this engine. This was the first step toward allowing MOST to support multiple spacecraft.

As of the time of writing this paper, the prototype Space Dynamics Engine is being developed into a complete spacecraft simulator, both for use as a standalone program, and embedded in a spacecraft OTB. Work is about to begin on a proper SED, and associated DNS. The team is also developing ideas for additional ways of bringing data to MOST, such as databases and direct network links. Future work on the data section of MOST will concentrate on these efforts.

On the display side, research on Qt has been completed. It has been found to have all the elements needed for a proper implementation of the current version of MOST. All team members are being trained in Qt, and the current prototype of MOST is being ported to that environment. As a bonus, the Qt research has revealed a feature of Qt that will allow for easy software configuration, which will be another step toward the use of MOST with multiple spacecraft.

B. Management and Schedule

The basic philosophy that will be used to manage the COSMOS/MOST development project is to use a core team of very experienced professionals to mentor graduate students, with support of undergraduate students. MOST development is being planned so that MOST will be operational and ready to support the HawaiiSat mission, which is scheduled for launch in late 2011 or early 2012. MOST must be ready at least six months before launch to support integration and test activities, mission planning, and operations personnel training (including rehearsals).

VI. Conclusion

MOST is primarily a visualization and commanding tool designed specifically for supporting real-time contact operations for one or more spacecraft, but due to its versatility, it can support many additional areas of mission operations including mission planning, simulations and testing, training and rehearsal, trending and analysis, and anomaly resolution. MOST enables a single Flight Controller to provide full monitoring and control capability from a single console with a user-configured GUI. It is designed to be easily adaptable to a new MOC and to be tailored for new spacecraft. This portability is aided by the use of the Qt programming environment under a Lesser GNU Public License, which allows source code to be available to qualified users without violating ITAR restrictions. The inclusion of a powerful “lights-out” capability in MOST makes it especially suitable for university and other small satellite operations conducted with limited resources of personnel and funding. MOST is being developed by the HSFL with the collaboration of NASA/ARC and SCU with the ultimate goal that COSMOS/MOST will be adopted as a standard for use by universities and other small spacecraft operators. MOST is currently in the prototype stage of development and is being designed initially to support operations at two MOCs (at HSFL and at NASA/ARC) of three small satellites to be launched in 2012.

Acknowledgments

The authors wish to thank Luke Flynn (Director, HSFL), Bruce Yost (NASA/ARC), John Hines (NASA/ARC), Chris Kitts (SCU), and Lavina Chatlani (HSFL) for their support of this project.

References

1. Sorensen, T.C., et al, "Effective Science Mission Planning and Operations - The Clementine Approach," Paper RAL.GS.31, 1st Annual Reducing the Cost of Space Ground Systems and Operations Symposium, Rutherford-Appleton Laboratories, June 1995.
2. Qt Technical Overview Whitepaper - http://qt.nokia.com/files/pdf/qt-4.4-whitepaper pg. 41).

.