SciBox: A Software Library for Rapid Development of Science Operation Simulation, Planning, and Command Tools

Teck H. Choo and Joseph P. Skura

SciBox is an APL-developed software library designed specifically for space operation simulation, planning, and commanding. It is not a ready-to-use application but rather a toolbox for rapid development of customized, focused software applications. It was built to meet the needs of several programs under development in the APL Space Department. The SciBox library is constructed with an extensible architecture that allows capabilities to be continually acquired and integrated into it. The ultimate goals are to enable rapid development of high-fidelity operation simulation tools for use at the earliest stages of mission development when critical instrument design decisions are made, and to reduce the cost of developing operation simulation tools for use in spacecraft testing, instrument testing, and science operation planning and commanding.

INTRODUCTION

Early requirements analysis is critical to many space missions, and it frequently requires simulation of the probable spacecraft and instrument operations. However, a high-fidelity Operation Simulation Tool (OST) takes substantial time to develop and can require great effort to complete. With mission development phases getting shorter, it is impractical to complete the development of an OST from the ground up in time for use in requirements analyses during the design phase of a spacecraft or instrument. Most missions will opt for the more practical solution of "good enough" rough estimates, but these estimates are occasionally inaccurate. When the inaccuracies are discovered late in the mission development cycle, it results in major design changes, cost increases, and schedule slips.

The challenge is to be able to develop an OST early, before these critical design decisions are made. The APL Space Department has built a software toolbox, SciBox, with the objective of enabling the rapid development of an OST. SciBox contains highly reusable software packages designed specifically for space operation simulation. It is built on an extensible architecture that allows continual acquisition and integration of new capabilities into the library for reuse in future missions.

The OST is also used for science operation planning during the space mission. Observation plans are

formulated by simulating various observation scenarios within the spacecraft's and instruments' constraints. These scenarios are designed to optimize the performance of the instruments and to maximize the science data that can be obtained without violating any constraints. The optimal observation scenario is then used to generate the command sequences that are sent to the spacecraft and the instruments. Recognizing that operation simulation is a critical part of operations planning and analysis, we have extended SciBox's design objectives to include support for the development of mission operation tools.

This article describes the motivation for creating SciBox, its layout, and its uses in spacecraft and instrument programs being built and flown by the APL Space Department.

MOTIVATING FACTORS

SciBox was inspired by the lessons learned from various Space Department programs. In October 2000, the size of the reaction wheels for MESSENGER (MErcury Surface, Space Environment, GEochemstry and Ranging) needed to be determined. MESSENGER is a three-axis stabilized spacecraft, and the reaction wheels are used to control the orientation and pointing of the spacecraft in space. The project office asked us to develop a simulation tool to model the spacecraft orientation based on spacecraft constraints and probable science observation scenarios in order to support the studies of momentum buildup in the reaction wheels.

At first, the simulation was specific to the analysis of the reaction wheel. However, it was later discovered that this initial OST was applicable to a variety of spacecraft and instrument requirements analyses. Initially it was adapted to aid the MDIS[1] (Mercury Dual Imaging System) instrument imaging strategy.[2] MESSENGER is to map the entire planet to an average of less than 250 m/pixel throughout the course of the mission. With tight orientation constraints and resources, we needed to demonstrate an imaging strategy that would satisfy these requirements. Next, the OST was modified to validate the heat flux that the MDIS instrument deck would receive while orbiting Mercury.[3] Unfortunately, the simulation results demonstrated that the original design could not meet the camera thermal requirements. A direct consequence of this discovery was a modification of the MDIS camera design.

While the OST for MESSENGER was under development, teams from other space programs such as CRISM[4] (Compact Reconnaissance Imaging Spectrometer for Mars), PEPSSI (Pluto Energetic Particle Spectrometer Science Investigation), TIMED (Thermosphere, Ionosphere, Mesosphere Energetics and Dynamics), and MIMI (Magnetospheric IMaging Instrument) were inquiring about similar tools. CRISM and PEPSSI were in the development phase, and their needs were similar to those of MESSENGER, where the OST was to be used to determine hardware and flight software capabilities that meet the science observation requirements.

- CRISM, which is to be flown on MRO (Mars Reconnaissance Orbiter), is a high-resolution hyperspectral imager, and its initial interest in the OST was to study both the spacecraft and instrument image motion compensation capabilities required to meet the imager's resolution requirements.
- PEPSSI is an energetic particle instrument to be flown onboard New Horizons, built by APL to fly by Pluto. The PEPSSI instrument has a 160° × 12° field of view (FOV), and it must be mounted on the spacecraft such that it can maximize the particle samples collected while avoiding direct solar illumination during the flyby of Pluto.

TIMED and MIMI were already in the operational phase, where use of the OST focuses on operation planning. The objective in this phase is to analyze the science observation opportunities given the instrument and flight software capabilities.

- TIMED (http://www.timed.jhuapl.edu/) is an Earth-orbiting satellite built by the APL Space Department. It carries four primary instruments: GUVI (Global Ultraviolet Imager), TIDI (TIMED Doppler Interferometer), SABER (Sounding of the Atmosphere using Broadband Emission Radiometer), and SEE (Solar Extreme Ultraviolet Experiment). The OST is used to help the ground stations plan for simultaneous observations of the atmosphere with GUVI, TIDI, and SABER.
- MIMI (http://sd-www.jhuapl.edu/CASSINI/) is a suite of three particle instruments built by APL and flown onboard the Jet Propulsion Laboratory's Cassini spacecraft. Cassini was launched in 1997 and arrived at Saturn on 1 July 2004. In orbit, the detector must be able to observe different regions defined by Saturn's magnetic field while trying to minimize the chances of dust particles from Saturn's rings entering the instruments. It must also keep the spacecraft radiator pointed at least 90° away from the Sun's direction.

Adding to MIMI's operation planning difficulties is that the planning and commanding steps are laborious manual processes requiring several iterations among the instrument scientists, the sequencer, the instrument engineer, and the mission operator. The process is usually initiated by the science team or investigating scientists specifying the type of science observations that need to be made. The instrument scientist then works with command sequencers to search for an observation opportunity and manually constructs the command sequence. The process is iterated several times between

the sequencers and the instrument scientists until a sequence satisfying the objectives is achieved. The final sequence is then sent to the mission operators for integration with other instrument command sequences and the spacecraft command sequence. As part of the integration process, the mission operators validate the instrument command sequence to ensure that it does not threaten the spacecraft's health or safety. The validation includes simulating the integrated sequence using an engineering model or software simulator. The instrument sequence must also be approved by the instrument engineer to ensure that it does not affect the health and safety of the instrument itself. Often, because of insufficient planning time, many great science opportunities are missed or incorrectly executed, resulting in the loss of valuable science observation time. Sometimes errors in the manually constructed sequences have even jeopardized the spacecraft.

Improvement in the efficiency and safety of the science operation could easily be gained by integrating the OST, command generation, and systematic validation of the command sequences. The instrument scientist could use the OST to find observation opportunities, and when one is selected, to automatically generate and validate the command sequences. However, the development of an integrated, efficient, operational tool involves multiple disciplines and requires a large software development effort.

Regardless, it is clear that the OST is crucial to many aspects of a space mission. The challenges are to make the OST available in the early phase of spacecraft and instrument development, when critical design decisions are made, and to reduce the cost of developing efficient, integrated operation tools.

LAYOUT

The diversity of space missions adds to the challenge of developing an OST in a short time and at reduced cost. There are different mission designs, i.e., orbiter versus flyby. The payloads each mission carries also vary greatly, i.e., *in situ* instruments such as particle and field instruments versus remote sensing instruments such as visible imagers. In addition, the environments in which these missions operate also differ. For example, at Mercury, MESSENGER's primary concern is heat exposure from the Sun as well as the radiation from Mercury. On the other hand, at Saturn, MIMI's concern is keeping the dust from damaging the instrument. The uniqueness of these missions and their unforeseen future needs imply that developing a generic OST that can be used by any mission is impractical.

The SciBox approach to addressing these challenges is through the application of software reuse technology. The idea is to build a software library to accumulate software components developed from existing programs. When a new program starts, instead of developing components from the ground up, components are chosen from the existing library and adapted for the unique mission requirements. When a new reusable software component is created for the new program, it is integrated into the software library. However, not all software components are equally reusable. To address this problem an internally developed design analysis technique is used to categorize these components as they are integrated back into SciBox. This design technique, the Hierarchical Component Design, was developed prior to SciBox to facilitate software reuse in the APL Space Department, and it has been successfully used in other data analysis projects. This technique is not covered in detail here, but a general overview follows to provide some background.

In the Hierarchical Component Design, software components are classified according to their degree of generality, as shown in Fig. 1. Software components with the highest generality are placed in the bottom layer; those in the layers above can use or reference any components below, but not vice versa. Components in the hierarchy must have no circular dependency. Software components designed specifically for an application are always placed in the top layer.

When the Hierarchical Component Design implementation was initially being developed, Java was selected as the foundation, and it was placed in the bottom layer. One of the design considerations in selecting Java was to take advantage of the rich library that comes with the Java Virtual Machine. This software library reduces the amount of development effort required to build any software application. Another consideration in selecting Java was the ability to run Java software on different computer platforms, so software developed could run on several different platforms.

Immediately above the Java Virtual Machine is the system-independent (SI) layer. This layer contains software components that are not available from the Java Virtual Machine but are frequently needed in data analysis–related software applications. The components are general in operation so they can be used by many different applications without the need for modification. Examples of software components in this layer are generic mathematical algorithms, array manipulation routines, and plotting routines. The SI layer was

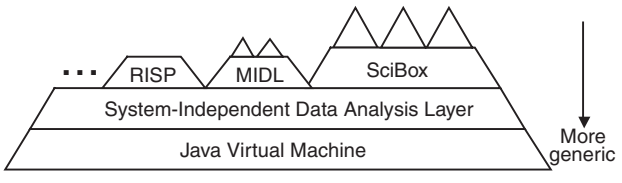

Figure 1. Software components are classified into layers depending on the degree of their generality (MIDL = Mission-Independent Data Layer, RISP = Air Force Real-Time Interplanetary Solar Proton Prediction system).

developed prior to SciBox and is being used to support a variety of data analysis applications in the Space Department. Examples are the Mission-Independent Data Layer for analyzing particle data observed by different spacecraft, the Air Force Real-Time Interplanetary Solar Proton Prediction system for monitoring space weather events, and biomedical projects.

SciBox lies on top of the SI layer and uses the data analysis components from that layer to build data analysis packages specific to space operation simulations but not specific to any particular space mission. Examples of SciBox software components are common mathematical algorithms used in celestial mechanics and astronomy, map projection, coordinate transformation, and scheduling and commanding. While the SciBox layer contains software that is independent of any specific mission, the software specific to a mission is located above SciBox in the project layer. The separation of the project-specific and the mission-independent layers in SciBox allows SciBox's software components to be reused in many projects. The robustness of the SciBox layer makes the project layer as thin as possible, which means that a minimal amount of work is required to complete this layer when SciBox is being adapted to new missions. Within SciBox, the software components are organized into four major sublayers using the Hierarchical Component Design. Those sublayers are shown in Fig. 2 and are described next.

The bottom layer of SciBox is the definition layer. Its purpose is to define the basic data structures frequently used in space operation simulations. These data structures serve as the basic information used by higher-level software components to communicate with each other. Examples of these data structures are coordinate system, body state, body shape, and instrument FOV. All of the data structures are implemented using Java classes and interfaces instead of simple basic variables such as integer or floating point. Such implementation takes advantage of the benefits offered by the strongly typed Java programming language. One advantage is that the wrong type of information cannot be passed from one component to another. Thus, the definition layer not only provides a clear definition of data structures being used in space operation simulations, but also implicitly makes any application developed more robust.

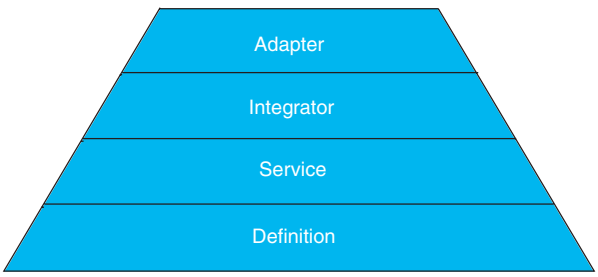

Figure 2. SciBox's major layers.

The layer just above SciBox's definition layer is the service layer, which captures frequently used, efficient algorithms commonly found in celestial mechanics and space physics. The service layer's components operate on the basic data structures defined in the lower definition layer to provide a service. Services include efficient computational algorithms for orbit propagation, geometry computation, state simulation, opportunity searching algorithm, spacecraft and instrument pointing control, schedule optimization, etc. These algorithms are designed to take advantage of the characteristics of orbital mechanics to improve computational performance and ease of use for application development.

In addition to algorithms, the service layer contains various information visualization packages covering two-dimensional (2D) and three-dimensional (3D) plots and displays. An example of 2D plots would be the ground track of a satellite or the simulated footprint of an instrument. The 3D plots include displays of a spacecraft's orientation above a celestial body or a body's terrain in 3D. The visualization of information is an essential component in the analysis of any multidimensional problem. Given that space operation simulations require solutions in a nonlinear N-dimensional space, analysis of these simulations requires the ability to visualize the results in many different cross-sectional views. The visualization components in this layer produce these different views. They are not only useful in providing a qualitative assessment of a simulation model but are also essential in code and model development. When a new simulation model is developed, the validation process for the model includes visualizing the results in various 2D and 3D plots.

Above SciBox's service layer is the integrator layer. Here, components from the service layer are integrated and packaged into a higher-level software component. Each of these integrated components provides a single service, but the analysis of space operation simulations requires many services. For example, when a planet-flyby geometry is being analyzed, the spacecraft pointing model and multiple visualization services are used to visualize the spacecraft attitude in 3D, its ground track, the instrument footprint, etc. The integrator layer is where these visualization services are integrated with a graphical user interface to provide an intuitive and easy-to-use interface for these services. Once one integrator component has been developed, the development time for future missions needing a similar component is greatly reduced. Most likely, the new mission only has to supply customized input parameters to the existing integrator component. If there is no suitable integrator component available, an existing integrator component similar to the one needed may be modified or extended without having to build the entire component from scratch. Besides integrating services, the integrator layer is also used to collect lessons learned from each mission.

These lessons learned are used to improve the simulation models, which in turn are used to advance a future, similar mission.

The topmost layer of SciBox is the adapter layer, which has a two-fold purpose. The first is to collect and reuse data input adapters from various externally developed data models and standards. For example, for planetary missions, spacecraft trajectories are usually provided in SPICE format. On the other hand, for an Earth-orbiting satellite, trajectories are derived from the NASA ASCII-formatted two-line elements. With these two adapters, either data format can be used directly with SciBox. The second purpose of this layer is to support application deployment. Different projects have different accessibility requirements for the developed tools. Some missions have very strict control on access to the operation center software, and only authorized personnel are allowed to use the tools on specific computers. An example would be the operation planning and commanding tools, which are available only from within a restricted area. Other missions may want easy access to a tool from anywhere in the world via the Internet, such as the TIMED coincidence calculator (http://www.timed.jhuapl.edu/coincidence_JTMD/). The adapter layer contains components that will configure the deployment environment for a developed tool. In summary, the focus of the adapter layer is not on the simulation but on user access and the configuration environment.

Choosing the user access environment, selecting services in the integrators, and specifying the input and output data formats are mission specific and are not part of SciBox. These customization efforts are handled at the project-specific level.

New components are continually being integrated into SciBox. With each new component added to the library, the work required for the next project is potentially reduced. Although much work remains, the software components accumulated in SciBox have begun to make major contributions to existing programs. In the next section, the use of SciBox components in two existing APL programs, MESSENGER and CRISM, is illustrated.

APPLICATIONS

MESSENGER Flyby OST

MESSENGER will have three Mercury flybys before orbital insertion. Even though the first flyby will not occur until 2007, operation planning has already begun. To analyze which observations are possible, an OST is being built to enable the MESSENGER science planning group to visualize the flyby from different views and control the orientation of the spacecraft.

In Fig. 3, the top left panel is a spacecraft 4π-FOV plot. It is the view from the spacecraft of the entire sky. The red rectangle is the projected MDIS camera FOV. The star field is obtained from the Position and Proper Motions (PPM) Astronomical Database Catalog from Goddard Space Flight Center. The top middle panel is the plot of the position and orbit of Earth, Venus, and Mercury at the particular simulated time. The top right panel allows the spacecraft attitude model to be selected, and it is used to control the orientation of the spacecraft. The bottom left plot is the 3D view of MESSENGER flying by Mercury. The white area on the surface of the planet in the 3D simulation has not been imaged before but will be imaged by MESSENGER. The bottom right plot is used to identify viewing opportunities for the various instruments as MESSENGER flies by Mercury. At the bottom is a "VCR-like" control panel that has "Play," "Stop," "Step forward," and "Step backward" buttons and a slider for moving the time step in the simulation. Most of the components displayed are selected directly from SciBox (Fig. 4).

The VCR play panel is selected from the SI layer because it is a generic component that was developed and used by other data analysis programs. The spacecraft 4π-FOV plot, planet orbit plot, 3D plot, and spacecraft attitude model are all selected from the SciBox service layer. The inputs to these plots are the Java interfaces defined in the definition layer. The definition layer only specifies the input data contents, but the actual formats of these contents are defined in the adapter layer. The flyby integrator combines all these services into one application, and it also provides an interface for projects to add new services. When the MESSENGER flyby

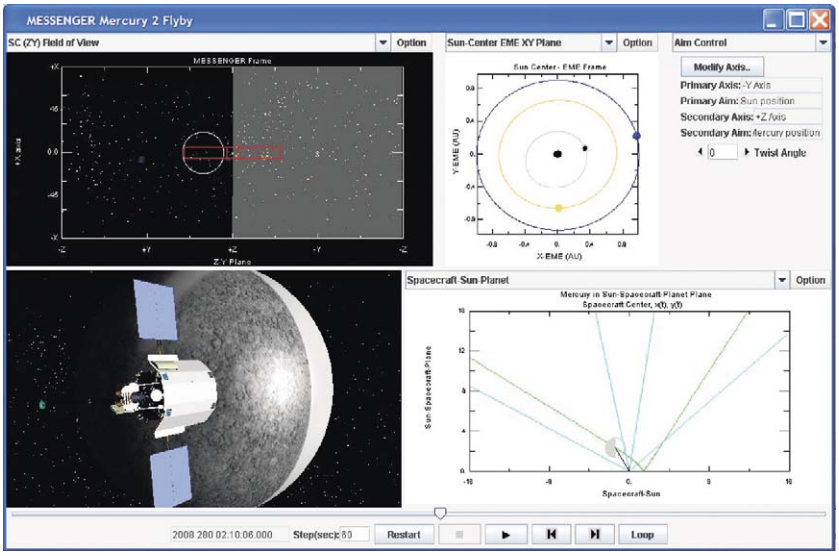

Figure 3. MESSENGER Mercury flyby analyzers.

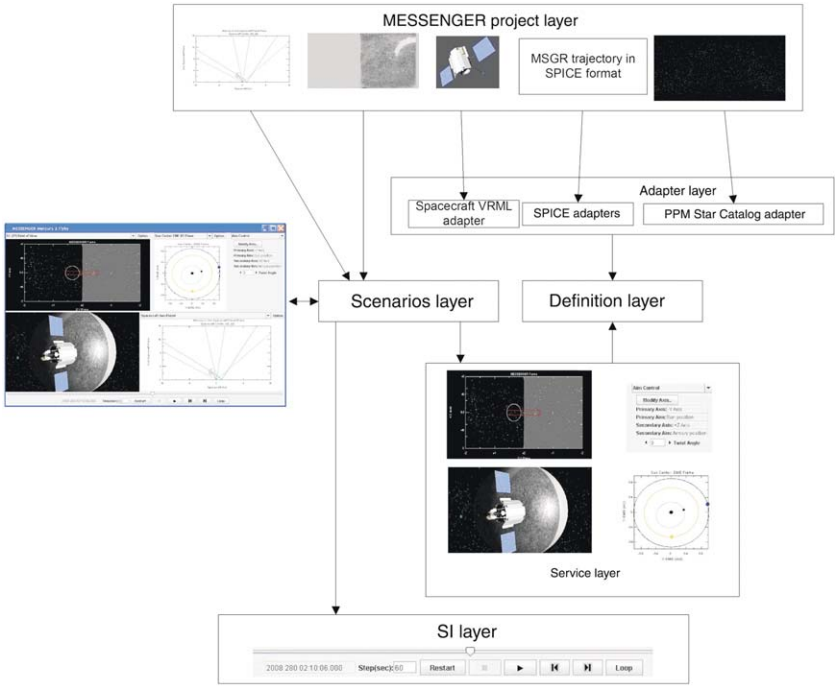

Figure 4. Construction of the MESSENGER flyby analyzer using SciBox components.

analyzer was created, most of the work was already done. All that was needed was to provide the actual inputs and to add any MESSENGER-specific visualization. In this case, the inputs were the 3D model of the spacecraft in VRML (Virtual Reality Modeling Language) format, the high-resolution Mercury map in simple cylindrical projection, and the ephemeredes of MESSENGER and planets in SPICE format.

CRISM Imager

Figures 5 through 8 demonstrate a Point-and-Command Operation System that is being developed for CRISM. The figures illustrate a series of steps from target selection, to opportunity search, to observation validation, to schedule integration, and, finally, to command generation.

Figure 5 is the CRISM target selector. The top left panel is the topography of Mars obtained from the MOLA (Mars Orbiter Laser Altimeter) projected in a simple cylindrical frame. Since the terrain map is enormous and cannot be displayed in the normal resolution of a computer monitor, the map has a zoom box that allows multiple levels of zooming. The enlarged image is orthographically projected and displayed in the top right panel. The bottom panel contains an existing list of areas of interest for science investigation. This list is expected to grow to more than 5000 entries by the time of operation in 2006.

When a target is selected, the opportunity analyzer searches the entire mission using the predicted spacecraft trajectory for all possible observation opportunities that meet the specified constraints. With current computer power and search algorithms selected from SciBox, the result is returned in seconds and is then displayed in an opportunity analyzer window. Shown in Fig. 6 is the opportunity analyzer for the CRISM off-nadir targeted observation mode. The opportunity analyzer is used to evaluate the candidates for observation. For the opportunities shown in Fig. 6, the best candidate usually is the first available opportunity with the lowest incident and spacecraft roll angle. When a candidate is selected, it can be visually evaluated and validated. Selecting an observation mode and one of the opportunities in Fig. 6 and then pressing the "Show Selection" button will open a new window illustrating a detailed simulation of the observation, shown in Fig. 7.

Scheduling the selected observation for commanding the spacecraft is as easy as pressing the "Add to Schedule" button in Fig. 6. Once pressed, the planned observation is automatically validated for constraint and

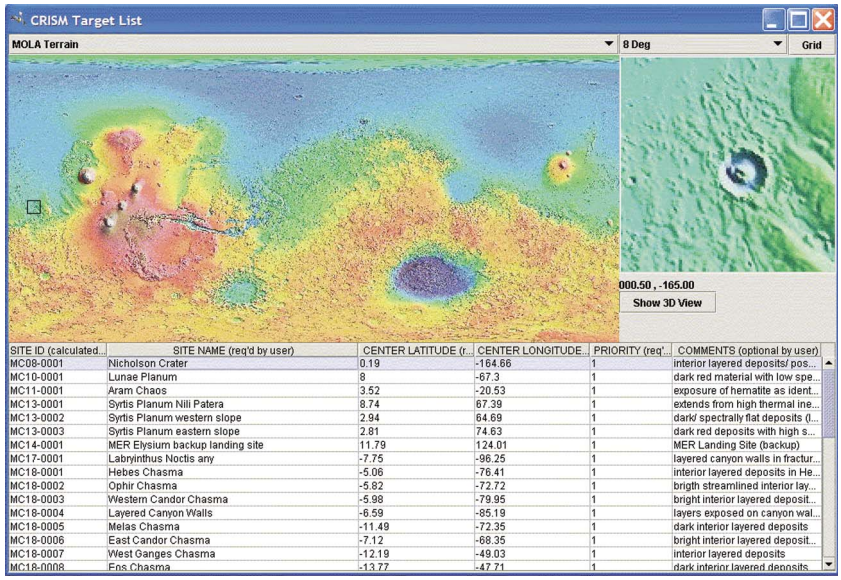

Figure 5. CRISM observation target list.

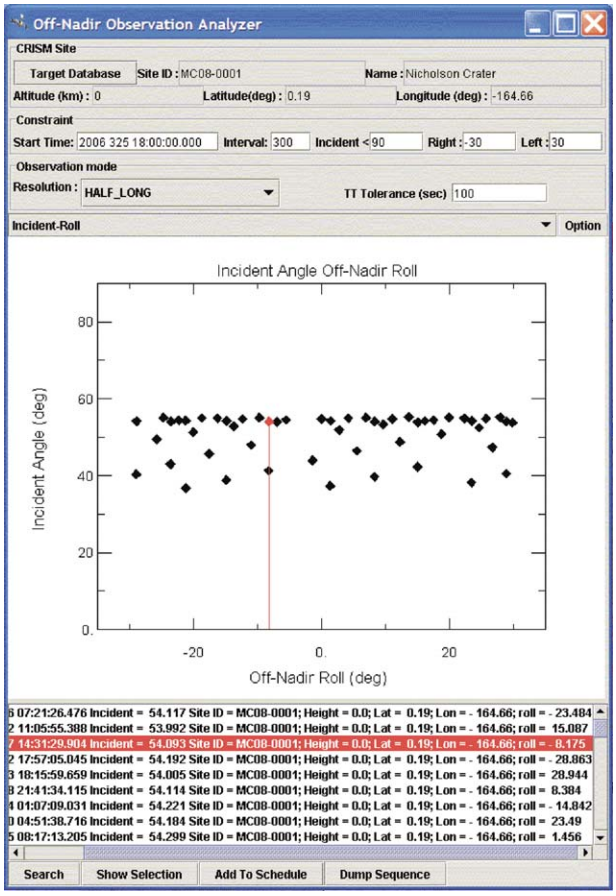

Figure 6. The CRISM off-nadir targeted observation opportunity analyzer displays potential observation opportunities that CRISM will likely have within the first 300 days of operation, with an incident angle of less than 90° and within the range of +30° to −30° spacecraft roll angles. Included in the search result are the predicted incident angle and the predicted spacecraft off-nadir roll angle required to point CRISM to the target.

schedule conflicts. If any conflict arises, the observation is rejected; otherwise, it is integrated into the observation schedule. The observation schedule can be viewed with the schedule editor (Fig. 8), which has multiple views of the observation (the view shown in Fig. 8 is the spatial distribution of the scheduled observations on the surface of Mars). It can be saved and loaded to continue previous work.

When observations are ready to be commanded, the "Generate Command" button is pressed, and the command sequence for all the scheduled observations is automatically validated and generated. Although CRISM has not yet been flown, the concept has been proven inside the laboratory. During testing and debugging of the instrument flight software, the Point-and-Command tool was used to generate flight-like command sequences for debugging the flight software and validating its performance. With the Point-and-Command System, operation simulation, opportunity searching, observation validation, and command generation being automated, the science team can focus more on the targets of interest and less on manually searching for opportunities and constructing command sequences.

During CRISM operation, it is expected that more than 5000 targets will be available in the database. Evaluating these targets every week for observation can be very tedious. The ultimate goal is to build a rule-based schedule optimization engine to automatically select targets from the database and schedule them based on their priorities, constraints, favorable viewing opportunities, and available resources. The observation schedule created by the optimizer can then be refined by the instrument scientist using the schedule editor.

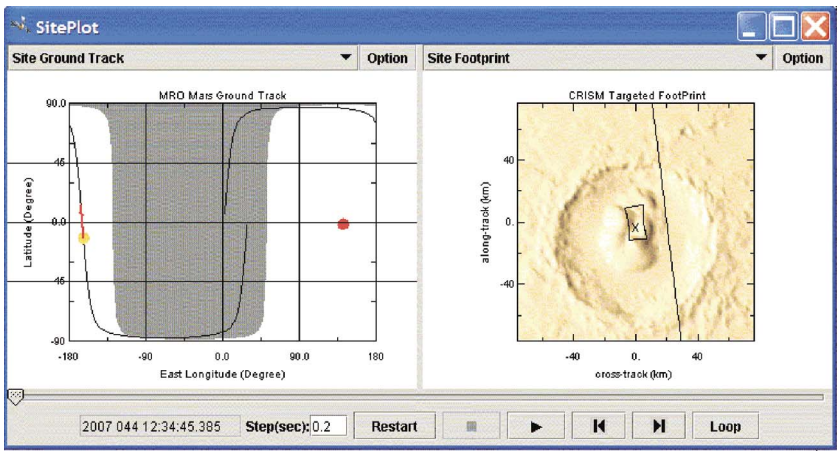

Figure 7. CRISM commanded observation simulator. The left plot shows the ground track of the Mars Reconnaissance Orbiter (MRO) and location where the observation will be made; the right plot shows the spacecraft ground track and footprint of the target of interest with simulated Mars session lighting angles. Many other views of the simulation are available from the selection menu.

FUTURE DEVELOPMENT

Given the increasing sophistication of missions, the development of a high-fidelity operation simulation becomes even more essential. Tools such as those built on SciBox will have to be used to assist in mission development and to improve efficiency during operation. To reap maximum benefit from SciBox, an operational simulation must be adopted in the early stages of development; the tool must be relatively complete before spacecraft and instrument integration and testing. Currently many space missions are launched without testing flight-like command sequences. Many of those command sequences are

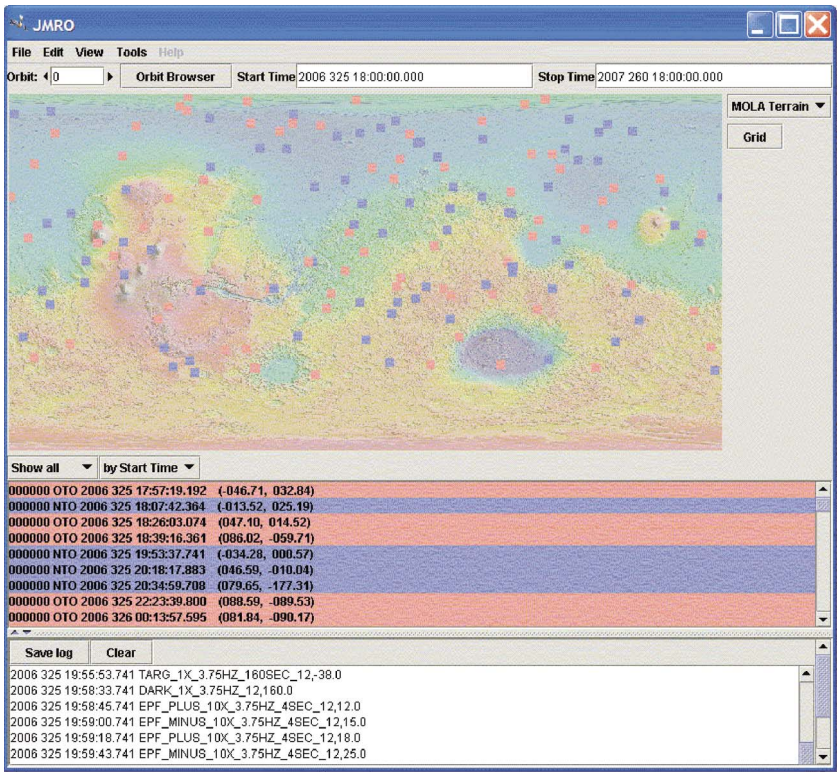

Figure 8. Schedule editor/integrator.

designed after the spacecraft is launched. In the future, the intention of SciBox is to make command sequence generation feasible during spacecraft and instrument integration and testing so that instruments delivered are fully tested and an efficient planning and commanding system is in place well before launch.

SUMMARY

Building and operating a spacecraft and its instruments are fundamentally complex and challenging problems. Operation simulation is one of the key components in untangling this complexity. With the unique nature of space programs, SciBox must cover multiple disciplines: software engineering, information visualization, *in situ* instruments, remote-sensing instruments, celestial mechanics, space physics, and operation commanding. All this is possible as a result of several factors. First, the availability of modern software technology, namely, Java and the Hierarchical Components Design technique, allows better management of software complexity and software reuse. Second, computation power has increased significantly over the years so that higher-fidelity simulations can be performed without using a supercomputer. Finally, and most important of all, is the open culture and team-oriented environment in the APL Space Department. The ability to work with instrument scientists from various disciplines, instrument engineers, and operational staff directly without red tape is a must.

REFERENCES

[1]Prockter, L. M., Robinson, M. S., Murchie, S. L., Bussey, D. B. J., Choo, T., et al., "The MESSENGER Mercury Dual Imaging System (MDIS): Imaging Strategy at Mercury," *Extraterrestrial Mapping Workshop*, Int. Soc. for Photogrammetry and Remote Sensing (Mar 2003).

[2]Choo, T. H., Murchie, S. L., and Jen, J. S., "The MESSENGER Science Planning Tool," *Mercury: Space Environment, Surface, and Interior*, The Field Museum, Chicago, IL (Oct 2001).

[3]Hawkins, S. E., Choo, T. H., and Ercol, C. J., *The MESSENGER/MDIS Thermal Environment: Verification of Spacecraft Heat Flux Models*, SRI-01-024, JHU/APL, Laurel, MD (Aug 2001).

[4]Murchie, S. L., Choo, T. H., et al., "CRISM: Compact Reconnaissance Imaging Spectrometer for Mars on The Mars Reconnaissance Orbiter," in *Proc. Lunar and Planetary Science Conf.*, Houston, TX (2002).

ACKNOWLEDGMENTS: We have been very fortunate to work with various leaders, particularly those in the SR Branch of APL's Space Department, who have fostered a team-oriented approach to solving challenges rather than a client–customer-oriented approach. In fact, when we decided to build the library, the term SciBox was coined by Deborah Domingue, the deputy project scientist of MESSENGER.

THE AUTHORS

TECK H. CHOO is a member of the Senior Professional Staff at APL. He has a B.S. and an M.S. in electrical engineering from the University of Kansas. Before coming to APL, he worked as a software engineer at Space Telescope Science Institute, where he had developed the data processing pipeline for the Hubble Space Telescope. Since joining APL, he has developed software tools for data analysis and visualization and simulation tools for Galileo, ACE, Ulysses, New Horizons, Cassini, TIMED, CRISM, and MESSENGER. He has been the principal investigator or co-investigator on several NASA Applied Information Research Program grants. Currently, he is the instrument operation lead for CRISM. His e-mail address is teck.choo@jhuapl.edu.

JOSEPH P. SKURA is a member of the Senior Professional Staff at APL. He has a B.S. and an M.S. in applied physics from Adelphi University. Since joining APL in 1978, he has been active in nonacoustic anti-submarine research, radar propagation, and biomedical research. He holds several patents in biomedical engineering for sensors for bone-healing, osteoporosis, and magneto-encephalography. After joining the Space Department in 1997, he used his scientific experience to develop software tools for scientific analysis, visualization, and simulations. He has developed applications for Galileo, NEAR, Auroral Particles and Imagery, TIMED, SSUSI, UPOS, New Horizons, and MESSENGER. His e-mail address is joseph.skura@jhuapl.edu.