OBJECT-ORIENTED TECHNOLOGIES IN A MULTI-MISSION DATA SYSTEM

Susan C. Murphy
Kevin J. Miller
John J. Louie

Operation Engineering Lab
Jet Propulsion Laboratory, 301-345
California Institute of Technology
Pasadena, California 91109-8099

ABSTRACT

The Operations Engineering Laboratory (OEL) at JPL is developing new technologies that can provide more efficient and productive ways of doing business in flight operations. Over the past three years, we have worked closely with the Multi-Mission Control Team to develop automation tools, providing technology transfer into operations and resulting in substantial cost savings and error reduction. The OEL development philosophy is characterized by object-oriented design, extensive reusability of code, and an iterative development model with active participation of the end users. Through our work, the benefits of object-oriented design have become apparent for use in mission control data systems.

In this paper, we will explain object-oriented technologies and how they can be used in a mission control center to improve efficiency and productivity. We will also discuss the current research and development efforts in the JPL Operations Engineering Laboratory to architect and prototype a new paradigm for mission control operations based on object-oriented concepts.

Key Words: Operations, Automation, Data Analysis, Object-Oriented

1. INTRODUCTION

The Multi-Mission Ground Data System (MGDS) at JPL has brought improvements and new technologies to mission operations. The development of a generic data system to meet the needs of multiple missions was intended to avoid re-inventing capabilities for each new mission and thus reduce costs. The traditional mainframe-based data systems of the past were expensive to modify and their proprietary architectures did not facilitate incorporation of new technologies. The MGDS is based on a distributed architecture, with powerful UNIX workstations, incorporating standards and open system architectures.

The MGDS is being expanded beyond its data delivery capabilities to include automation and analysis tools for the more demanding missions of the future. However, automation tools can help reduce costs only if they are focused on the people and the tasks they perform. New technologies may only bring minimal cost savings if the new system functions much like the old one. This often happens since the users who write the requirements aren't always familiar with the capabilities of new technologies and simply use their existing system as a model. For example, the mission controllers asked for a scrolling screen that displayed telemetry values representing the latest value of the spacecraft clock. This was the way the old system allowed them to determine whether there were any data outages. The developers gave them their scrolling display and operators continued to stare at these displays watching for outages. An important opportunity was lost to automate this process and improve the efficiency of operations. To solve these types of communications problems, a new approach was tried. Each division was assigned responsibility for its end-to-end system, from development through operations. In response, the Operations Engineering Lab (OEL) was created several years ago to merge operations and development activities for the Space Flight Operations Section.

2. OPERATIONS ENGINEERING LAB

2.1 Development Approach

Over the past several years, the OEL developers have worked closely with mission controllers and spacecraft engineers to develop a set of graphical automation tools. The tools utilize object-oriented graphics and direct-manipulation interfaces. In traditional ground data systems, data presentation and data access are not intuitive. Specialized languages must be learned by the user in order to describe the way data must be processed, accessed, and displayed. An object-oriented approach can simplify the user's interaction with the data system by modeling the system as made up of objects, entities defined by their functional and inherited characteristics. Object-oriented paradigms are ideal for developing easy-to-use graphical user interfaces where data and functions can be activated and manipulated directly on the screen.

Our approach has been successful because we build tools that are integrated, application-specific, and focused on automating essential, yet tedious and time consuming, operations tasks. In addition, we involve users and trainers early in the development process. In fact, we have mission operators work as developers in the lab, sometimes on a part-time basis and, in other cases, full-time for a limited tenure. Conversely, four of the OEL developers worked as members of the Spacecraft and Mission Control Teams at JPL and the Spacecraft Anomaly Team at the Cape in support of the recent Mars Observer launch. This has allowed us to maintain close contact with our users and understand the problems that need to be solved.

We develop software incrementally, as a series of rapid protoflight models that are reviewed constantly by the users. Their active participation has meant that the new technologies have been accepted more readily, and even more importantly, it has often made them enthusiastic to learn a new system. We have found that it is very important to get protoflight implementations in the hands of users and trainers as soon as possible to provide quick feedback to developers and to gain user acceptance. Our protoflight models have evolved to fully-implemented subsystems of the MGDS that have produced significant cost savings in operations. Similar successes are occurring at other NASA centers as the advances in workstations and open architectures have enabled small programming teams to quickly and cheaply implement automation tools that expand their capabilities [1].

2.2 SEG: An Object-Oriented Development Success Story

One of the initial projects in the OEL was to automate the Sequence of Events Generation (SEG) process that develops the detailed schedules and instructions for the ground control of a spacecraft. The inputs to the SEG process include the spacecraft command sequence, ground resource allocations, and special ground events. The output products include a text listing of the events (the Sequence of Events (SOE)) and a timeline display (the Space Flight Operations Schedule (SFOS)). [2] In the past, much of the SEG process was manual, fragmented, and understood by only a few operators. The mainframe-based software for SOE generation was expensive to maintain and modify and the interface was difficult and usually tedious. The SFOS timeline product was produced on a PC with a word processor. The process was slow, error-prone, and inefficient in the use of personnel. Editing a document was cumbersome, as only a small portion of the page was visible at once. Printing was done on a line printer, and the output was reduced and copied for distribution across the world. The frequent updates to these documents quickly invalidated the user's copy. The hard-copy products made it difficult for users to isolate the events of interest from the thousands of SOE items.

2.3 Building a Multi-Mission SEG

The SEG process was separated into two parts in order to isolate the mission-specific software that was expected to change frequently from the more stable, multi-mission graphical tools that would be needed

to display, edit, view, and print the SOE and SFOS. The mission-specific software are the scripts that automatically generate the ASCII data files that are then input to the SFOS and SOE graphical tools. Both graphical tools were designed to be generic viewing/editing tools with no hard-coded knowledge of the meaning of its contents. As a result, only the scripting software needs to be adapted to produce schedules for different missions and for different application areas. For example, the SFOS editor has been used to produce timeline displays of other schedules such as command sequences and mission planning schedules, simply by changing the file generation scripts.

2.4 Object-Oriented Design

The graphical SFOS editing/viewing tool was designed to make the operator's job fast and easy. The entire SFOS page is visible on the screen, with *What-You-See-Is-What-You-Get* capabilities for viewing, editing, and printing. We used an Object-Oriented (O-O) design approach in which each item can be directly manipulated on-screen as a graphics object.

In an object-oriented design, the system is designed around the **data** that the system must manipulate rather than focusing on the **functions** a system must perform. Objects are defined by their functional characteristics; they encapsulate knowledge of both their current state and expected behavior. Embedded in each object is an 'understanding' of its attributes and the methods it will use for performing its allowed functions. In an O-O system, data objects are often designed to model real-world objects. For example, in the SFOS editor, each object represents a ground or spacecraft event that will occur for a mission. An object that is a spacecraft command belongs to a different class of objects than one that describes a tracking activity.

In the SFOS editor, each object knows how to display, edit, delete, add, and move itself. For example, schedule objects will automatically place themselves in the correct timeline position. When a user selects an object to edit, the object will respond by bringing up its unique edit dialog box which displays items that are specific to that class of objects. The values displayed in each item reflect the current state values for that particular instance of the object. The menu-driven interface allows a user to change the timeline scale on-the-fly.

The graphics display is saved in an ASCII file which completely describes the format and content of the SFOS. Each line in the ASCII file corresponds to an object on the screen. The standard ASCII file interface was chosen because it allowed the generation of the input data files to be automated using simple scripting languages such as Perl and AWK.

The SOE graphical tool has many of the same features as the SFOS tool. The SOE tool displays a time-ordered, column-formatted display of the SOE file. It has capabilities for searching on selectable criteria, filtering out events of interest, and highlighting events. The contents and format of the columns can be reconfigured by the user.

2.5 Templates for Operations Processes

Although the process was automated, there were many steps in the procedure and decisions had to be made on execution parameters based on a complicated data flow. It was clear that an interface was necessary to simplify the operator's job. An interface builder (OELSHELL) was first implemented for building graphical templates. The template builder was an extension of JPL's D. Smythe's Widget Creation Library which uses a resource file to configure the interface. The templates provide buttons that can be used to call a program and the output can be redirected as needed. File widgets allow a user to designate input files for the process. On-line help and arrows help describe the process.

Templates were built for the SEG process as shown in Figure 1. The SEG template shell allows a user to select which output products are desired and which input sources are needed, eliminating the need for users to know which *program* must be executed in each case. Templates have been built for

other processes and have proved valuable as interfaces for integrating multiple utilities and for use as an operations training tool.

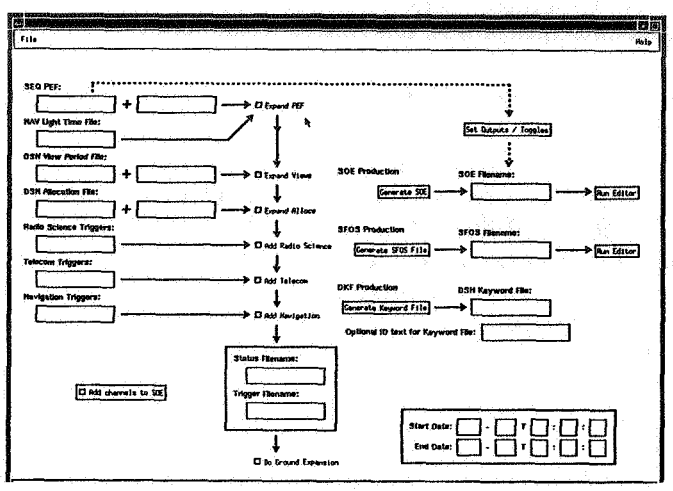

FIGURE 1. SEG TEMPLATE SHELL

The graphical SEG tools have been used as the basis for other editors and operations support tools. The editor/viewers were originally implemented using the SunView windowing environment. With the lab's migration to the new MGDS environment, our software had to be translated into the X/Motif windowing environment. The task was greatly simplified because of the object-oriented approach and the migration was done ahead of schedule and under budget. The system design did not have to change in the new environment, only the user-interaction functions were affected. Our original software design also included a lower-level set of graphics routines for drawing objects on the screen or to a laser printer. Only these routines had to be changed to interface with the X-server. Since all of our software tools used the same graphics routines, our development costs were significantly reduced through this commonality.

2.6 OEL Automation and Analysis Tools

Other automation tools developed in the OEL include an automated telemetry log generator for data management. This on-line log tool has replaced the previous method of hand-written logs describing telemetry data coverage. An alarm clock tool provides user-programmable clocks and graphics in a workstation window. Another program provides an interactive window-based editing tool for generating milestone schedule charts. A Multi-Mission Smart Alarm tool was built to monitor data outages on MGDS broadcast circuits, eliminating the need for controllers to manually detect outages by watching a scrolling time display. Several other real-time monitoring tools were developed to automatically analyze and log information on engineering packet telemetry, telemetry data gaps, telecommunication uplink frequency verification, and command execution verification. Object-oriented analysis tools have been built, including graphical tools for browsing through telemetry channel information and generating statistical summaries, and plotting channel data. The graphical plotting tool provides sophisticated capabilities for plotting multiple channels versus time or channel versus channel plots. The ASCII file interface has allowed telemetry plots to be produced automatically by generating the input files with simple scripting tools that supply hooks into the MGDS telemetry database retrieval system.

A decommutation map tool provides a graphical means of visualizing the telemetry decommutation process and of interactive editing of a channel object. The tool allows a user to graphically navigate through a decommutation map, moving in and out of sub-commutations using a mouse. The current mode for manipulating decommutation maps consists of editing a file written in a specialized map language. In order to make a change to a specific map, an operator must understand the decommutation map language and must also read through the lengthy source code logic. Undoubtedly, an intuitive, graphical method for visual display and editing of decommutation maps is necessary.

2.7 Results

The software interfaces to the graphical tools are designed to be clean and simple. This has enabled our software to be used for multiple missions with differing objectives. These tools have resulted in substantial savings and reduced errors. The SFOS editor resulted in

a two-thirds reduction in work force required to operate the system and eliminated costly mainframe processing costs. The ASCII file interface and its electronic availability has given scientists access to up-to-date schedules, eliminating printing and shipping expenses that amounted to up to $200K per project per year. The Smart Alarm Tool has saved up to 40% of the mission controllers time that used to be spent monitoring broadcast lines for data outages. The other automated monitoring tools have eliminated many error-prone, time-consuming manual processes.

It is evident that increased automation of the operations process is necessary and that a careful strategy is required to ensure that the tools focus on specific tasks and that the approach is flexible enough to meet the needs of multiple missions. In addition, the O-O approach has resulted in less costly maintenance, and we have been able to accommodate users' and missions' changing needs with minimal expense.

We have also discovered that developing great tools is not enough. It is essential for developers to get into the operational environment and assist operations teams in adapting the delivered system and its many tools to meet their needs. The OEL has led a Customer Adaptation Team (CAT) for adapting the MGDS for the Voyager and Mars Observer Spacecraft and Mission Control Teams. This effort has been very successful because we work in their environment, configuring the workstations on their desks, building scripts to automate their tasks, and designing interfaces to integrate tools.

3. ADVANCED RESEARCH

3.1 Object-Oriented Operations (O^3)

Another emerging trend in the development of data analysis and display software is the use of modular software components (or tools) that are integrated and manipulated by the user as *objects* in a desktop environment. For example, in a ground data system, the software components would implement operations processes, each with inputs, outputs, and a knowledge base. The objects would closely model the way one thinks about the data system. The objects would understand and account for the rules that govern its behavior and would navigate through the complex layers in the data system itself. The components could be interactively strung together to establish a flow of data according to a user-defined process.

We have proposed an Object-Oriented Operations (O^3) system environment that incorporates these techniques in a mission control data system. In a fully-integrated O^3 environment, the objects could be invoked by events that occur on the ground or on the spacecraft. For example, if the spacecraft unexpectedly went into safe mode, the ground data system would detect the event and signal appropriate objects (telemetry channels) to be instantiated, producing the necessary displays or analysis of their current status on the screen for visual inspection by the engineer.

In an initial O^3 prototype, the OEL is investigating the combination of expert system and object-oriented technologies. Using the Gensym G2 real-time object-oriented expert system, we have built a model of the Mars Observer telecommunications system including the ground system that is driven by the SOE. The spacecraft and each ground station are represented by an icon on a top-level schematic, and the overall state of the system is displayed by changing colors of parts of these icons. Each icon has a specific sub-workspace which can display more specific information about each component of the system. Each object has associated rules that allow the system to compare and verify the state of the spacecraft with respect to the state of the ground tracking stations.

3.2 Object-Based Interaction Paradigm

We are also investigating extending object-oriented concepts to a system environment which provides a mechanism for the tools themselves to communicate through an object-based interaction. The system

interface would allow a user to *drag* objects between components using a mouse. The objects would carry functional information with them, allowing each process to respond to an object in its own unique way. The object-based interaction paradigm has presented some interesting research challenges, and a prototype version has been implemented. In the prototype system, the graphical decommutation map tool and a hypertext dictionary tool interact through common telemetry channel objects. The user can select a channel object in the map and then drag it to the dictionary tool for information on the channel (or vice versa). The object activates different functions in the different tools, enabling a user to focus on the data they want to access and display, rather than on the tools and methods needed to get the job done.

3.3 Closed-Loop Monitoring System

Another OEL research project is an automated closed-loop monitoring system that provides real-time integration of uplink events with downlink telemetry information, using the SOE as the predict source. In the existing SOE for Mars Observer, each spacecraft command item has a descriptive text field that contains a list of related downlink telemetry channels. These channels are monitored in real-time by mission controllers using another tool that reads the downlink telemetry stream. The closed-loop system will integrate these tasks by interfacing the SOE with the real-time telemetry data stream and automatically appending appropriate channel values with command items. To integrate downlink events, the SOE will require a new type of channel data object that is treated as a special field and will be configured to send messages to external processes that will monitor the appropriate channels. Although the channels selected for viewing will be automated, a user may opt to monitor any channel at a given time, simply by adding that channel object at the appropriate point in the time-ordered SOE listing.

This integrated system will greatly simplify the user's ability to access and view telemetry data, and will provide a means to view this data in the context of the commands and predicted values that are used to interpret it. It also provides a mechanism to automate generation of mission controller logs, *as-flown* SOEs, and dynamic alarms. This tool will also prove very useful in ground support equipment for spacecraft integration and test.

4. CONCLUSION

With new development approaches such as that of JPL's MGDS and Operations Engineering Laboratory, success has been shown in improving mission operability and reducing cost in operations. The use of object-oriented technologies has resulted in software that is easier to use and cheaper to adapt for multiple missions and changing mission objectives. The future of mission control at JPL is one of opportunity and continued improvement. Work at the OEL continues to make use of these opportunities to improve productivity in mission control.

5. ACKNOWLEDGEMENTS

This work was done at the Jet Propulsion Laboratory, California Institute of Technology, under a contract from the National Aeronautics and Space Administration. We would like to acknowledge the work of the technical staff in the OEL, especially Patrick Curran, Roger Davidson, Ana Maria Guerrero, Joseph Hu, Daniel Hurley, Chester Joe, and Christine Aguilera. We would also like to acknowledge the JPL Mission Operations Teams for their enthusiasm and support.

6. REFERENCES

1. Muratore, J., 1991. Real-Time Data System at Space Shuttle Mission Control, *NASA Control Center Conference.*

2. K. Miller and S. Murphy, 1990. Sun Technology for Mission Operations, *Sun Tech Journal.*