by use of the OGC Style Layer Descriptor (SLD) protocol. Full-precision spectral arithmetic processing is also available, by use of a custom SLD extension. This server can dynamically add shaded relief based on the Lunar elevation to any image layer. This server also implements tiled WMS protocol and super-overlay KML for high-performance client application programs.

This program was written by Lucian Plesea of Caltech and Trent Hare of the United States Geological Survey for NASA's Jet Propulsion Laboratory.

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-45951.

Expressions Module for the Satellite Orbit Analysis Program

The Expressions Module is a software module that has been incorporated into the Satellite Orbit Analysis Program (SOAP). The module includes an expressions-parser submodule built on top of an analytical system, enabling the user to define logical and numerical variables and constants. The variables can capture output from SOAP orbital-prediction and geometric-engine computations. The module can combine variables and constants with built-in logical operators (such as Boolean AND, OR, and NOT), relational operators (such as >, <, or =), and mathematical operators (such as addition, subtraction, multiplication, division, modulus, exponentiation, differentiation, and integration). Parentheses can be used to specify precedence of operations.

The module contains a library of mathematical functions and operations, including logarithms, trigonometric functions, Bessel functions, minimum/maximum operations, and floating-point-to-integer conversions. The module supports combinations of time, distance, and angular units and has a dimensional-analysis component that checks for correct usage of units. A parser based on the Flex language and the Bison program looks for and indicates errors in syntax. SOAP expressions can be built using other expressions as arguments, thus enabling the user to build analytical trees. A graphical user interface facilitates use.

This program was developed by Robert Carnright, David Stodden, Jim Paget, and John Coggi of Caltech for NASA's Jet Propulsion Laboratory.

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-45052.

Virtual Satellite

Virtual Satellite (VirtualSat) is a computer program that creates an environment that facilitates the development, verification, and validation of flight software for a single spacecraft or for multiple spacecraft flying in formation. In this environment, enhanced functionality and autonomy of navigation, guidance, and control systems of a spacecraft are provided by a virtual satellite — that is, a computational model that simulates the dynamic behavior of the spacecraft.

Within this environment, it is possible to execute any associated software, the development of which could benefit from knowledge of, and possible interaction (typically, exchange of data) with, the virtual satellite. Examples of associated software include programs for simulating spacecraft power and thermal-management systems. This environment is independent of the flight hardware that will eventually host the flight software, making it possible to develop the software simultaneously with, or even before, the hardware is delivered. Optionally, by use of interfaces included in VirtualSat, hardware can be used instead of simulated. The flight software, coded in the C or C++ programming language, is compilable and loadable into VirtualSat without any special modifications. Thus, VirtualSat can serve as a relatively inexpensive software test-bed for development test, integration, and post-launch maintenance of spacecraft flight software.

This program was written by Stephan R. Hammers of the Hammers Co., Inc. for Goddard Space Flight Center. Further information is contained in a TSP (see page 1). GSC-14824-1

Small-Body Extensions for the Satellite Orbit Analysis Program (SOAP)

An extension to the SOAP software allows users to work with tri-axial ellipsoid-based representations of planetary bodies, primarily for working with small, natural satellites, asteroids, and comets. SOAP is a widely used tool for the visualization and analysis of space missions. The small body extension provides the same visualization and analysis constructs for use with small bodies. These constructs allow the user to characterize satellite path and instrument cover information for small bodies in both 3D display and numerical output formats.

Tri-axial ellipsoids are geometric shapes the diameters of which are different in each of three principal *x*, *y*, and *z* dimensions. This construct provides a better approximation than using spheres or oblate spheroids (ellipsoids comprising two common equatorial diameters as a distinct polar diameter). However, the tri-axial ellipsoid is considerably more difficult to work with from a modeling perspective. In addition, the SOAP small-body extensions allow the user to actually employ a plate model for highly irregular surfaces. Both tri-axial ellipsoids and plate models can be assigned to coordinate frames, thus allowing for the modeling of arbitrary changes to body orientation.

A variety of features have been extended to support tri-axial ellipsoids, including the computation and display of the spacecraft sub-orbital point, ground trace, instrument footprints, and swathes. Displays of 3D instrument volumes can be shown interacting with the ellipsoids. Longitude/latitude grids, contour plots, and texture maps can be displayed on the ellipsoids using a variety of projections. The distance along an arbitrary line of sight can be computed between the spacecraft and the ellipsoid, and the coordinates of that intersection can be plotted as a function of time. The small-body extension supports the same visual and analytical constructs that are supported for spheres and oblate spheroids in SOAP making the implementation of the more complex algorithms largely transparent to the user.

This work was done by Robert Carnright of Caltech and David Stodden and John Coggi of The Aerospace Corporation for NASA's Jet Propulsion Laboratory.

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-45054.

Scripting Module for the Satellite Orbit Analysis Program (SOAP)

This add-on module to the SOAP software can perform changes to simulation objects based on the occurrence of specific conditions. This allows the software to encompass simulation response of

scheduled or physical events. Users can manipulate objects in the simulation environment under programmatic control. Inputs to the scripting module are Actions, Conditions, and the Script. Actions are arbitrary modifications to constructs such as Platform Objects (i.e. satellites), Sensor Objects (representing instruments or communication links), or Analysis Objects (user-defined logical or numeric variables). Examples of actions include changes to a satellite orbit (*v*), changing a sensor-pointing direction, and the manipulation of a numerical expression. Conditions represent the circumstances under which Actions are performed and can be couched in If-Then-Else logic, like performing *v* at specific times or adding to the spacecraft power only when it is being illuminated by the Sun.

The SOAP script represents the entire set of conditions being considered over a specific time interval. The output of the scripting module is a series of events, which are changes to objects at specific times. As the SOAP simulation clock runs forward, the scheduled events are performed. If the user sets the clock back in time, the events within that interval are automatically undone.

This script offers an interface for defining scripts where the user does not have to remember the vocabulary of various keywords. Actions can be captured by employing the same user interface that is used to define the objects themselves. Conditions can be set to invoke Actions by selecting them from pull-down lists. Users define the script by selecting from the pool of defined conditions. Many space systems have to react to arbitrary events that can occur from scheduling or from the environment. For example, an instrument may cease to draw power when the area that it is tasked to observe is not in view. The contingency of the planetary body blocking the line of sight is a condition upon which the power being drawn is set to zero. It remains at zero until the observation objective is again in view. Computing the total power drawn by the instrument over a period of days or weeks can now take such factors into consideration. What makes the architecture especially powerful is that the scripting module can look ahead and behind in simulation time, and this temporal versatility can be leveraged in displays such as *x–y* plots. For example, a plot of a satellite's altitude as a function of time can take changes to the orbit into account.

This work was done by Robert Carnright of Caltech and David Stodden, John Coggi, and Jim Paget of The Aerospace Corporation for NASA's Jet Propulsion Laboratory.

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-45055.

XML-Based SHINE Knowledge Base Interchange Language

The SHINE Knowledge Base Interchange Language software has been designed to more efficiently send new knowledge bases to spacecraft that have been embedded with the Spacecraft Health Inference Engine (SHINE) tool. The intention of the behavioral model is to capture most of the information generally associated with a spacecraft functional model, while specifically addressing the needs of execution within SHINE and Livingstone. As such, it has some constructs that are based on one or the other.

As NASA/JPL autonomous science missions go deeper and deeper into space, the collection of unexpected data becomes a problem. Data structures can easily be implemented in advance that can collect any kind of data; however, when it comes to processing the data into information and taking advantage of serendipitous science discovery, designing a fixed and efficient data structure becomes increasingly complex. This software defines and implements a new kind of data structure that can be used for representing information that is derived from serendipitous data discovery. It allows the run-time definition of arbitrarily complex structures that can adapt at run-time as the raw science data is transformed into information.

This solves the problem decision trees can be prone to, namely how expensive they can be to execute because of the need to evaluate each non-leaf node and, based upon its truth, to either progress deeper into the structure or to examine an alternative. This requires many machine cycles, which can negatively affect time-critical decisions.

This software runs on a variety of different platforms, including SUN, HP, Intel, Apple Macs, Flight Processors, etc. It can be distributed in either source code or binary code and requires a LISP compiler to run with a number, such compilers being either commercially available or found as shareware. The software has no specific memory requirements and depends on the applications that are running in it. It is implemented as a library package and folds into whatever environment is calling it.

Currently, this software is a component of the Common Automation Engine (CAE) that was developed for Deep Space Network (DSN). It has been in active use for over three years and has been installed in a shadow mode running at Goldstone and DSN monitoring operations at JPL.

This work was done by Mark James, Ryan Mackey, and Raffi Tikidjian of Caltech for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-44546.

Core Technical Capability Laboratory Management System

The Core Technical Capability Laboratory Management System (CTCLMS) consists of dynamically generated Web pages used to access a database containing detailed CTC lab data with the software hosted on a server that allows users to have remote access. Users log into the system with their KSC (or other domain) username and password. They are authenticated within that domain and their CTCLMS user privileges are then authenticated within the system. Based on the different user's privileges (roles), menu options are displayed. CTCLMS users are assigned roles such as Lab Member, Lab Manager, Natural Neighbor Integration Manager, Organizational Manager, CTC Program Manager, or Administrator. The role assigned determines the users' capabilities within the system. Users navigate the menu to view, edit, modify or delete laboratory and equipment data, generate financial and managerial reports, and perform other CTC lab-related functions and analyses.

High availability and detail of lab data gives management insight into the needs and requirements of KSC CTC-funded labs. Comprehensive, quantitative, current data are available in one easily accessible location for Program Operating Plan (POP) development, justification of POP submittals, overguide requests, contract renewals, and phasing of maintenance and replacement requirements. Lab health is quantitatively understandable. Financial and managerial reports are generated automatically from detailed data, and facilitate uniform com-