Software

Program Synthesizes UML Sequence Diagrams

A computer program called “Rational Sequence” generates Universal Modeling Language (UML) sequence diagrams of a target Java program running on a Java virtual machine (JVM). Rational Sequence thereby performs a reverse engineering function that aids in the design documentation of the target Java program. Whereas previously, the construction of sequence diagrams was a tedious manual process, Rational Sequence generates UML sequence diagrams automatically from the running Java code. Moreover, there is no need to insert instrumentation code into the target Java program. Rational Sequence employs the Java Native Interface application programming interface to create a software profiler that plugs into the JVM. Once the user starts the target Java program, Rational Sequence acts as a nonintrusive observer, generating UML diagrams representing the observed activity. Every method call, object instantiation, or thread event of the target Java program is tracked by the profiler. Once the Java program has ended, the profiler generates a UML model that contains packages, classes, and all method calls observed during the execution of the target program. The user can control the way the UML model is generated by specifying packages and/or classes to be included in the diagrams.

This program was written by Matthew R. Barry and Richard N. Osborne of United Space Alliance for **Johnson Space Center**. *For further information, contact the Johnson Technology Transfer Office at (281) 483-3809.*
MSC-23656

Aspect-Oriented Subprogram Synthesizes UML Sequence Diagrams

The Rational Sequence computer program described in the immediately preceding article includes a subprogram that utilizes the capability for aspect-oriented programming when that capability is present. This subprogram is denoted the Rational Sequence (AspectJ) component because it uses AspectJ, which is an extension of the Java programming language that introduces aspect-oriented programming techniques into the language. The Rational Sequence (AspectJ) component is compiled with a target Java application program on an AspectJ compiler. The user then starts the Java application program. Thereafter, the Rational Sequence (AspectJ) component publishes every visible method call to a Universal Modeling Language (UML) sequence diagram. When the Java application program ends, a sequencer proceeds to generate a UML model that contains packages, classes, and all method calls that occurred during the execution of the program. The user can control the way the UML model is generated by specifying, via the aspect source code, packages and/or classes to be included in the diagrams. Like the rest of Rational Sequence, the AspectJ component complies with the UML specification.

This program was written by Matthew R. Barry and Richard N. Osborne of United Space Alliance for **Johnson Space Center**. *For further information, contact the Johnson Technology Transfer Office at (281) 483-3809.*
MSC-23655

Updated Computational Model of Cosmic Rays Near Earth

An updated computational model of the galactic-cosmic-ray (GCR) environment in the vicinity of the Earth, Earth’s Moon, and Mars has been developed, and updated software has been developed to implement the updated model. The GCR model and software in their original forms, developed during the early 1990s, were based on balloon and satellite data from 1954 to 1992. This model accounts for solar modulation of the cosmic-ray contribution for each element from hydrogen through iron by computationally propagating the local interplanetary spectrum of each element through the heliosphere. The propagation is effected by solving the Fokker-Planck diffusion, convection, energy-loss boundary-value problem. Since August 1997, the Advanced Composition Explorer NASA satellite has provided new data on GCR energy spectra. These new data were used to update the original model and greatly improve the accuracy of prediction of interplanetary GCR. The updated software was also simplified significantly, relative to the original software. The updated model and software are expected to provide highly accurate GCR-environment data for use by interplanetary-mission planners in planning for protecting astronauts against radiation and ensuring radiation hardness of electronic equipment.

This program was written by Patrick M. O’Neill of **Johnson Space Center**. *For further information, contact the Johnson Technology Transfer Office at (281) 483-3809.*
MSC-23891

Software for Alignment of Segments of a Telescope Mirror

The Segment Alignment Maintenance System (SAMS) software is designed to maintain the overall focus and figure of the large segmented primary mirror of the Hobby-Eberly Telescope. This software reads measurements made by sensors attached to the segments of the primary mirror and from these measurements computes optimal control values to send to actuators that move the mirror segments. The software also acts as a logger for the collected data, a server from which the hardware of the control computer can acquire control information and other computers can collect data, and a monitoring and diagnostic system. The software provides a graphical user interface through which human operators can exert control. The software supports four modes of operation:

- *Operate* — The server acquires the sensory data and processes them into commands for the actuators.
- *Calibrate* — Calibration tests are performed on the edge sensors and the relationships between actuator commands and sensor responses are quantified.
- *Standby* — The server is initialized in standby mode, from which it can

make the transition to any of the other three modes.

- *Diagnostic* — This mode provides access to all sensory data in real time and is intended for use in diagnosis of sensor anomalies.

This program was written by Drew P. Hall, Richard T. Howard, William C. Ly, John M. Rakoczy, and John M. Weir of **Marshall Space Flight Center**. *For further information, contact Jim Dowdy, Commercialization Project Lead, at Jim.Dowdy@nasa.gov.*
MFS-31835-1

Simulation of Dropping of Cargo With Parachutes

Decelerator System Simulation (DSS) is a computer program for predicting and analyzing the dynamics of a load of cargo dropped with parachutes from an aircraft. A DSS simulation runs from the first motion in the aircraft until the payload reaches the ground. Intended for use in support of airdrop tests for the X-38 program, DSS was developed by modifying and augmenting an older program, denoted UD233A, used for simulating the dynamics of a space-shuttle solid rocket booster falling with a parachute. The main effort in converting UD233A into DSS involved development of computational models for simulating the inflation of one or more parachute(s), the dynamics of the payload and the slings connecting the parachute(s) with the payload, and the extraction of the payload and parachutes from the aircraft.

This program was written by Peter Cuthbert of **Johnson Space Center**. *For further information, contact the Johnson Technology Transfer Office at (281) 483-3809.*
MSC-2363

DAVE-ML Utility Programs

DAVEtools is a set of Java archives (*.jar files) that embodies tools for manipulating flight-dynamics models that have been encoded in dynamic aerospace vehicle exchange markup language (DAVE-ML). [DAVE-ML is an application program, written in Extensible Markup Language (XML), for encoding complete computational models of the dynamics of aircraft and spacecraft. The goal in the continuing development of DAVE-ML is to expedite the exchange and validation of dynamical models, via the Internet, in a manner that is consistent and is independent of computational-simulation facilities, computing languages, and simulation software.] At present, *DAVEtools* includes two tools:

- *dave* (a basic DAVE-ML parser), which generates a Java-based version of a model encoded in DAVE-ML and
- *dave2sl*, which builds on *dave* to create Simulink® representations of models encoded in DAVE-ML.

The manipulations that can be performed at the present early stage of development are rather limited. More importantly, *DAVEtools* serves as an example of how to write an import software tool for a DAVE-ML file.

This program was written by Bruce Jackson of **Langley Research Center**. *For further information, access http://daveml.nasa.gov.*
LAR-16879-1

Robust Control for the Mercury Laser Altimeter

Mercury Laser Altimeter Science Algorithms is a software system for controlling the laser altimeter aboard the Messenger spacecraft, which is to enter into orbit about Mercury in 2011. The software will control the altimeter by dynamically modifying hardware inputs for gain, threshold, channel-disable flags, range-window start location, and range-window width, by using ranging information provided by the spacecraft and noise counts from instrument hardware. In addition, because of severe bandwidth restrictions, the software also selects returns for downlink. To reduce mission risk, the software incorporates three different modes of operation. The three modes are denoted as fixed, range-driven, and closed-loop (or adaptive). The fixed mode provides fixed hardware inputs for all but the threshold. The range-driven mode receives and utilizes ranging information from the spacecraft regarding its slant range to the planet or asteroid. The adaptive mode is capable of improving upon the ranging information provided by the spacecraft by use of a closed-loop range-estimation algorithm. The software is sufficiently robust that it could be used on other missions, and in fact, this has already been proposed.

This program was written by Jacob S. Rosenberg of **Goddard Space Flight Center**. *Further information is contained in a TSP (see page 1).*
GSC-14876-1