These programs provide this reachability information in an easy-to-use format by combining the surface position and orientation, arm kinematics, instrument mounting, and instrument approach angles. This software is also integrated into the ground data system and the automated processing pipelines. It understands the EDR and RDR file formats and metadata, and products tailored for in situ surface operations.

This work was done by Robert G. Deen, Patrick C. Leger, Matthew L. Robinson, and Robert G. Bonitz of Caltech for NASA's Jet Propulsion Laboratory. For more information, contact iaoffice@jpl.nasa.gov.

This software is available for commercial licensing. Please contact Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-47731.

JPL Space Telecommunications Radio System Operating Environment

A flight-qualified implementation of a Software Defined Radio (SDR) Operating Environment for the JPL-SDR built for the CoNNeCT Project has been developed. It is compliant with the NASA Space Telecommunications Radio System (STRS) Architecture Standard, and provides the software infrastructure for STRS compliant waveform applications. This software provides a standards-compliant abstracted view of the JPL-SDR hardware platform. It uses industry standard POSIX interfaces for most functions, as well as exposing the STRS API (Application Programming Interface) required by the standard. This software includes a standardized interface for IP components instantiated within a Xilinx FPGA (Field Programmable Gate Array).

The software provides a standardized abstracted interface to platform resources such as data converters, file system, etc., which can be used by STRS standards conformant waveform applications. It provides a generic SDR operating environment with a much smaller resource footprint than similar products such as SCA (Software Communications Architecture) compliant implementations, or the DoD Joint Tactical Radio Systems (JTRS).

This work was done by James P. Lux, Minh Lang, Kenneth J. Peters, Gregory H. Taylor, Courtney B. Duncan, David S. Orozco, Ryan A. Stern, Earl R. Ahten, and Mike Girard 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 Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-47766.

RFI-SIM: RFI Simulation Package

RFI-SIM simulates the RFI environment to estimate the interference from terrestrial emitters into spacecraft, or vice versa. A high-fidelity simulation of the RFI environment has been developed by employing all antenna-related and radar system-related parameters of multiple emitters, as well as that of the desired spacecraft.

In the simulation, the real-time analysis of the interference and its effects on error budgets of a desired radar system is taken into account. This provides a reliable tool for radar system design to deal with RFI issues and to evaluate the sensitivity of various parts of a radar system including antenna pattern, RF front-end and digital processing to RFI signals.

The simulator is capable of a high-fidelity, complex, and real-time simulation of RFI environment. It is flexible enough to be employed for various scenarios and for several NASA missions. RFI-SIM can perform the following in support of radar system design and performance analyses:

- Error budget analyses due to RFI on a space-borne radar system;
- Sensitivity analysis of the various radar parameters, as well as hardware specs, in the presence of RFI;
- Verification of the radar system design at several stages of RF and digital components in order to evaluate their robustness against RFI;
- Assistance in algorithm development for RFI detection and removal approach;
- Based on the available database, the RFI environment over North America at L-band has been reliably and successfully simulated and validated so it can be used for L-band space-borne radars in the RFI environment; and
- Estimation of the interference from space-borne radars into terrestrial FAA radars regarding FAA compatibility issues.

This work was done by Hirad Ghaemi and Curtis W. Chen 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 Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-48565.

ION Configuration Editor

The configuration of ION (Interplanetary Overlay Network) network nodes is a manual task that is complex, time-consuming, and error-prone. This program seeks to accelerate this job and produce reliable configurations.

The ION Configuration Editor is a model-based smart editor based on Eclipse Modeling Framework technology. An ION network designer uses this Eclipse-based GUI to construct a data model of the complete target network and then generate configurations. The data model is captured in an XML file. Intrinsic editor features aid in achieving model correctness, such as field fill-in, type-checking, lists of valid values, and suitable default values. Additionally, an explicit "validation" feature executes custom rules to catch more subtle model errors. A "survey" feature provides a set of reports providing an overview of the entire network, enabling a quick assessment of the model's completeness and correctness. The "configuration" feature produces the main final result, a complete set of ION configuration files (eight distinct file types) for each ION node in the network.

This work was done by Richard L. Borgen of Caltech for NASA's Jet Propulsion Laboratory. For more information, contact iaoffice@jpl.nasa.gov.

This software is available for commercial licensing. Please contact Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-48209.

Dtest Testing Software

This software runs a suite of arbitrary software tests spanning various software languages and types of tests (unit level, system level, or file comparison tests). The dtest utility can be set to automate periodic testing of large suites of software, as well as running individual tests. It supports distributing multiple tests over multiple CPU cores, if available.

The dtest tool is a utility program (written in Python) that scans through a directory (and its subdirectories) and finds all directories that match a certain pattern (directory name starts with "test_" or "test-") and then executes any tests in that directory as described in simple configuration files. The tests are completely arbitrary and are not tied to any specific programming language. A variety of tests is available to support comparing test output files