mand Message Files) on demand. It also enables developers to change the underlying database, while maintaining the same interface to the existing applications. The logging capabilities are also beneficial to operators when they are trying to recall how they solved a similar problem many days ago: this software enables automatic recovery of SCMF and RML (Robot Markup Language) sequence files directly from the command EVRs, eliminating the need for people to find and validate the corresponding sequences.

To address the lack of auditing capability for sequences onboard a spacecraft during earlier missions, extensive logging support was added on the Mars Science Laboratory (MSL) sequencing server. This server is responsible for generating all MSL binary SCMFs from RML input sequences. The sequencing server logs every SCMF it generates into a MySQL database, as well as the high-level RML file and dictionary name inputs used to create the SCMF. The SCMF is then indexed by a hash value that is automatically included in all command EVRs by the onboard flight software. Second, both the binary SCMF result and the RML input file can be retrieved simply by specifying the hash to a Restful web interface. This interface enables command line tools as well as large sophisticated programs to download the SCMF and RMLs on-demand from the database, enabling a vast array of tools to be built on top of it. One such command line tool can retrieve and display RML files, or annotate a list of EVRs by interleaving them with the original sequence commands.

This software has been integrated with the MSL sequencing pipeline where it will serve sequences useful in diagnostics, debugging, and situational awareness throughout the mission.

This work was done by Thomas W. Starbird, John R. Morris, Khawaja S. Shams, and Mark W. Maimone 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-48080.

MER Telemetry Processor

MERTELEMPROC processes telemetered data in data product format and generates Experiment Data Records (EDRs) for many instruments (HAZCAM, NAVCAM, PANCAM, microscopic imager, Mössbauer spectrometer, APXS, RAT, and EDLCAM) on the Mars Exploration Rover (MER). If the data is compressed, then MERTELEMPROC decompresses the data with an appropriate decompression algorithm. There are two compression algorithms (ICER and LOCO) used in MER. This program fulfills a MER specific need to generate Level 1 products within a 60-second time requirement.

EDRs generated by this program are used by merinverter, marscahv, marsrad, and marsjplstereo to generate higher-level products for the mission operations. MERTELEPROC was the first GDS program to process the data product. Metadata of the data product is in XML format. The software allows user-configurable input parameters, per-product processing (not stream-based processing), and fail-over is allowed if the leading image header is corrupted. It is used within the MER automated pipeline.

MERTELEMPROC is part of the OPGS (Operational Product Generation Subsystem) automated pipeline, which analyzes images returned by in situ spacecraft and creates level 1 products to assist in operations, science, and outreach.

This work was done by Hyun H. Lee 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-47797.

pyam: Python implementation of YaM

pyam is a software development framework with tools for facilitating the rapid development of software in a concurrent software development environment. pyam provides solutions for development challenges associated with software reuse, managing multiple software configurations, developing software product lines, and multiple platform development and build management. pyam uses release-early, release-often development cycles to allow developers to integrate their changes incrementally into the system on a continual basis. It facilitates the creation and merging of branches to support the isolated development of immature software to avoid impacting the stability of the development effort. It uses modules and packages to organize and share software across multiple software products, and uses the concepts of link and work modules to reduce sandbox setup times even when the code-base is large. One side-benefit is the enforcement of a strong module-level encapsulation of a module's functionality and interface. This increases design transparency, system stability, and software reuse.

pyam is written in Python and is organized as a set of utilities on top of the open source SVN software version control package. All development software is organized into a collection of "modules." pyam "packages" are defined as sub-collections of the available modules. Developers can set up private sandboxes for module/package development. All module/package development takes place on private SVN branches. High-level pyam commands support the setup, update, and release of modules and packages. Released and pre-built versions of modules are available to developers. Developers can tailor the source/link module mix for their sandboxes so that new sandboxes (even large ones) can be built up easily and quickly by pointing to pre-existing module releases. All inter-module interfaces are publicly exported via links. A minimal, but uniform, convention is used for building modules.

This work was done by Steven Myint and Abhinandan Jain 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-48447.