ing priorities between rules. A formulation and scenario for how cross-domain interoperability can be achieved have been developed based on a negotiation mechanism between different parties (domains) so that all parties can agree on procedures for interacting with each other. An implementation of this methodology has been developed in the form of an executable code and corresponding GUI interface.

The network management and Web communication software used by the different organizations presents a stumbling block. Many of the tools used by the various divisions do not have the ability to communicate network management data with each other. At best, this means that manual human intervention into the communication protocols used at various network routers and endpoints is required. This process is tedious, error-prone, and slow.

The present methods have inherent inefficiency and are not fully automatic, which heavily restricts their practical applications. The new method is based on an efficient algorithm. The new engine utilizes defeasible logic to describe communication policy constraints and priorities. Defeasible logic (see figure) is non-monotonic, and contains three different types of rules: strict rules, which are strict "if/then" statements; defeasible rules that are "if this, then probably that" statements; and defeater rules that contradict the outcomes of defeasible rules.

The policy negotiation program reads in two files specifying the policies that the user wishes to combine, and outputs a single file describing the means of communication that satisfy both input policies, if any can be found.

To implement this method, a tool called DPC (Defeasible Policy Composition) was developed. To maintain that efficiency in the DPC tool, the data structures for the individual terms of each constraint are joined in linked-list fashion to their constraints and to a parent object representing each term. This can be visualized as a linked grid, where the heads of each column are the terms, the heads of each row are the rule names, and the body of the grid is the references to the terms that make up those rules. Each term reference is linked to its neighbors in the grid, which allows the algorithm to quickly and efficiently search through, add, and delete rows, terms, and individual term references.

This work was done by Farrokh Vatan and Edward T. Chow of Caltech for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

The software used in this innovation is available for commercial licensing. Please contact Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-48399.

Linked-List-Based Multibody Dynamics (MBDyn) Engine

Lyndon B. Johnson Space Center, Houston, Texas

This new release of MBDyn is a software engine that calculates the dynamics states of kinematic, rigid, or flexible multibody systems. An MBDyn multibody system may consist of multiple groups of articulated chains, trees, or closed-loop topologies. Transient topologies are handled through conservation of energy and momentum. The solution for rigid-body systems is exact, and several configurable levels of nonlinear term fidelity are available for flexible dynamics systems.

The algorithms have been optimized for efficiency and can be used for both non-real-time (NRT) and real-time (RT) simulations. Interfaces are currently compatible with NASA's Trick Simulation Environment. This new release represents a significant advance in capability and ease of use. The two most significant new additions are an application programming interface (API) that clarifies and simplifies use of MBDyn, and a link-list infrastructure that allows a single MBDyn instance to propagate an arbitrary number of interacting groups of multibody topologies.

MBDyn calculates state and state derivative vectors for integration using an external integration routine. A Trick-compatible interface is provided for initialization, data logging, integration, and input/output.

This work was done by John Maclean, Thomas Brain, Leslie Quiocho, An Huynh, and Tushar Ghosh of Johnson Space Center. Further information is contained in a TSP (see page 1). MSC-24925-1

Multi-Mission Power Analysis Tool (MMPAT) Version 3

NASA's Jet Propulsion Laboratory, Pasadena, California

The Multi-Mission Power Analysis Tool (MMPAT) simulates a spacecraft power subsystem including the power source (solar array and/or radioisotope thermoelectric generator), bus-voltage control, secondary battery (lithium-ion or nickel-hydrogen), thermostatic heaters, and power-consuming equipment. It handles multiple mission types including heliocentric orbiters, planetary orbiters, and surface operations. Being parametrically driven along with its user-programmable features can reduce or even eliminate any need for software modifications when configuring it for a particular spacecraft. It provides multiple levels of fidelity, thereby fulfilling the vast majority of a project's power simulation needs throughout the lifecycle. It can operate in a standalone mode with a graphical user interface, in batch mode, or as a library linked with other tools.

This software can simulate all major aspects of a spacecraft power subsystem. It is parametrically driven to reduce or eliminate the need for a programmer. Added flexibility is provided through user-designed state models and table-driven parameters.

MMPAT is designed to be used by a variety of users, such as power subsystem engineers for sizing power subsystem components; mission planners for