In the multi-core system, the FTM resides on a single, dedicated core, separate from the cores used by the application. This is done in order to isolate the FTM from application faults and to allow it to swap out any application core for a substitute. The structure of the FTM consists of an interface to a fault tolerant strategy module, a responder module, a fault manager module, an error factory, and an error mapper that determines the severity of the error.

In the present reference implementation, the only fault tolerant strategy implemented is introspection. The introspection code waits for an application node to send an error notification to it. It then uses the error factory to create an error object, and at this time, a severity level is assigned to the error. The introspection code uses its built-in knowledge base to generate a recommended response to the error. Responses might include ignoring the error, logging it, rolling back the application to a previously saved checkpoint, swapping in a new node to replace a bad one, or restarting the application. The original error and recommended response are passed to the top-level fault manager module, which invokes the response. The responder module also notifies the introspection module of the generated response. This provides additional information to the introspection module that it can use in generating its next response. For example, if the responder triggers an application rollback and errors are still occurring, the introspection module may decide to recommend an application restart.

This work was done by Raphael R. Some, Paul L. Springer, Hans P. Zima, Mark James, and David A. Wagner of Caltech for NASA’s Jet Propulsion Laboratory. For more information, contact tiaoffice@jpl.nasa.gov.

This software is available for commercial licensing. Please contact Daniel Binderich of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-47976.

**Trick Simulation Environment 07**

The Trick Simulation Environment is a generic simulation toolkit used for constructing and running simulations. This release includes a Monte Carlo analysis simulation framework and a data analysis package. It produces all auto documentation in XML. Also, the software is capable of inserting a malfunction at any point during the simulation. Trick 07 adds variable server output options and error messaging and is capable of using and manipulating wide characters for international support. Wide character strings are available as a fundamental type for variables processed by Trick.

A Trick Monte Carlo simulation uses a statistically generated, or predetermined, set of inputs to iteratively drive the simulation. Also, there is a framework in place for optimization and solution finding where developers may iteratively modify the inputs per run based on some analysis of the outputs. The data analysis package is capable of reading data from external simulation packages such as MATLAB and Octave, as well as the common comma-separated values (CSV) format used by Excel, without the use of external converters. The file formats for MATLAB and Octave were obtained from their documentation files, is created in the simulation directory. Trick 07 provides an ASCII format. The binary option is capable of delivering the simulation data to multiple clients using multicast sockets. This allows multiple machines to receive the same data without increasing the computational load on the simulation.

In addition to Linux and MacOSX, Trick 07 now supports three real-time operating systems: QNX, LynxOS, and RedHawk Linux. Each RTOS has unique system calls accessing real-time features such as setting process priorities, processor assignment, and accessing the real-time clock. Trick uses the unique real-time features of each OS.

This work was done by Alexander S. Lin of Johnson Space Center and John M. Penn, Dan A. Strauss, and Keith Vetter of L-3 Communications Corporation. Further informa-