Verifying Diagnostic Software

Livingstone PathFinder (LPF) is a simulation-based computer program for verifying autonomous diagnostic software. LPF is designed especially to be applied to NASA's Livingstone computer program, which implements a qualitative-model-based algorithm that diagnoses faults in a complex automated system (e.g., an exploratory robot, spacecraft, or aircraft). LPF forms a software test bed containing a Livingstone diagnosis engine, embedded in a simulated operating environment consisting of a simulator of the system to be diagnosed by Livingstone and a driver program that issues commands and faults according to a nondeterministic scenario provided by the user. LPF runs the test bed through all executions allowed by the scenario, checking for various selectable error conditions after each step. All components of the test bed are instrumented, so that execution can be single-stepped both backward and forward. The architecture of LPF is modular and includes generic interfaces to facilitate substitution of alternative versions of its different parts. Altogether, LPF provides a flexible, extensible framework for simulation-based analysis of diagnostic software; these characteristics also render it amenable to application to diagnostic programs other than Livingstone.

This program was written by Tony Lindsey and Charles Pecheur of **Ames Research Center**. *Further information is contained in a TSP (see page 1).*

Inquiries concerning rights for the commercial use of this invention should be addressed to the Patent Counsel, Ames Research Center, (650) 604-5104. Refer to ARC-14780-1.

Initial Processing of Infrared Spectral Data

The Atmospheric Infrared Spectrometer (AIRS) Science Processing System is a collection of computer programs, denoted product generation executives (PGEs), for processing the readings of the AIRS suite of infrared and microwave instruments orbiting the Earth aboard NASA's Aqua spacecraft. Following from level 0 (representing raw AIRS data), the PGEs and their data products are denoted by alphanumeric labels (1A, 1B, and 2) that signify the successive stages of processing. Once level-0 data have been received, the level-1A PGEs begin processing, performing such basic housekeeping tasks as ensuring that all the Level-0 data are present and ordering the data according to observation times. The level-1A PGEs then perform geolocation-refinement calculations and conversions of raw data numbers to engineering units. Finally, the level-1A data are grouped into packages, denoted granules, each of which contain the data from a six-minute observation period. The granules are forwarded, along with calibration data, to the Level-1B PGEs for processing into calibrated, geolocated radiance products. The Level-2 PGEs, which are not yet operational, are intended to process the level-1B data into temperature and humidity profiles, and other geophysical properties.

This program was written by Solomon De Picciotto, Albert Chang, Zi-Ping Sun, Yuan-Ti Ting, Evan Manning, Steven Gaiser, Bjorn Lambrigtsen, Mark Hofstadter, Thomas Hearty, Thomas Pagano, Hartmut Aumann, and Steven Broberg 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 Don Hart of the California Institute of Technology at (818) 393-3425. Refer to NPO-35243.

Activity-Centric Approach to Distributed Programming

The first phase of an effort to develop a NASA version of the Cybele software system has been completed. To give meaning to even a highly abbreviated summary of the modifications to be embodied in the NASA version, it is necessary to present the following background information on Cybele:

Cybele is a proprietary software infrastructure for use by programmers in developing agent-based application programs [complex application programs that contain autonomous, interacting components (agents)]. Cybele provides support for event handling from multiple sources, multithreading, concurrency control, migration, and load balancing. A Cybele agent follows a programming paradigm, called activity-centric programming, that enables an abstraction over system-level thread mechanisms. Activity-centric programming relieves application programmers of the complex tasks of thread management, concurrency control, and event management. In order to provide such functionality, activity-centric programming demands support of other layers of software. This concludes the background information.

In the first phase of the present development, a new architecture for Cybele was defined. In this architecture, Cybele follows a modular service-based approach to coupling of the programming and service layers of software architecture. In a service-based approach, the functionalities supported by activity-centric programming are apportioned, according to their characteristics, among several groups called services. A well-defined interface among all such services serves as a path that facilitates the maintenance and enhancement of such services without adverse effect on the whole software framework. The activity-centric application-program interface (API) is part of a kernel. The kernel API calls the services by use of their published interface. This approach makes it possible for any application code written exclusively under the API to be portable for any configuration of Cybele.

This program was written by Renato Levy, Goutam Satapathy, and Jun Lang of Intelligent Automation, Inc., for **Johnson Space Center**. *For further information, contact:*

Intelligent Automation, Inc.
2 Research Place, Suite 202
Rockville, MD 20850

Refer to MSC-23239.

Controlling Distributed Planning

A system of software implements an extended version of an approach, denoted shared activity coordination (SHAC), to the interleaving of planning and the exchange of plan information among organizations devoted to different missions that normally communicate infrequently except that they need to collaborate on joint activities and/or the use of shared resources. SHAC enables the planning and scheduling systems of the organizations to coordinate by resolving conflicts while optimizing local planning solutions. The present software provides a framework for modeling and executing communication

protocols for SHAC. Shared activities are represented in each interacting planning system to establish consensus on joint activities or to inform the other systems of consumption of a common resource or a change in a shared state. The representations of shared activities are extended to include information on (1) the role(s) of each participant, (2) permissions (defined as specifications of which participant controls what aspects of shared activities and scheduling thereof), and (3) constraints on the parameters of shared activities. Also defined in the software are protocols for changing roles, permissions, and constraints during the course of coordination and execution.

This program was written by Bradley Clement and Anthony Barrett 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 Don Hart of the California Institute of Technology at (818) 393-3425. Refer to NPO-40438