Software

Mission Data System Java Edition Version 7

The Mission Data System framework defines closed-loop control system abstractions from State Analysis including interfaces for state variables, goals, estimators, and controllers that can be adapted to implement a goal-oriented control system. The framework further provides an execution environment that includes a goal scheduler, execution engine, and fault monitor that support the expression of goal network activity plans. Using these frameworks, adapters can build a goal-oriented control system where activity coordination is verified before execution begins (plan time), and continually during execution. Plan failures including violations of safety constraints expressed in the plan can be handled through automatic re-planning.

This version optimizes a number of key interfaces and features to minimize dependencies, performance overhead, and improve reliability. Fault diagnosis and real-time projection capabilities are incorporated. This version enhances earlier versions primarily through optimizations and quality improvements that raise the technology readiness level.

Goals explicitly constrain system states over explicit time intervals to eliminate ambiguity about intent, as compared to command-oriented control that only implies persistent intent until another command is sent. A goal network scheduling and verification process ensures that all goals in the plan are achievable before starting execution. Goal failures at runtime can be detected (including predicted failures) and handled by adapted response logic. Responses can include plan repairs (try an alternate tactic to achieve the same goal), goal shedding, ignoring the fault, cancelling the plan, or safing the system.

This work was done by William K. Reinholtz and David A. Wagner 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 Dan Broderick at Daniel.F.Broderick@jpl.nasa.gov. Refer to NPO-47804.

Adaptive Distributed Environment for Procedure Training (ADEPT)

ADEPT (Adaptive Distributed Environment for Procedure Training) is designed to provide more effective, flexible, and portable training for NASA systems controllers. When creating a training scenario, an exercise author can specify a representative rationale structure using the graphical user interface, annotating the results with instructional texts where needed. The author's structure may distinguish between essential and optional parts of the rationale, and may also include "red herrings" — hypotheses that are essential to consider, until evidence and reasoning allow them to be ruled out.

The system is built from pre-existing components, including Stottler Henke's SimVentive™ instructional simulation authoring tool and runtime. To that, a capability was added to author and exploit explicit control decision rationale representations. ADEPT uses SimVentive's Scalable Vector Graphics (SVG)-based interactive graphic display capability as the basis of the tool for quickly noting aspects of decision rationale in graph form.

The ADEPT prototype is built in Java, and will run on any computer using Windows, MacOS, or Linux. No special peripheral equipment is required.

The software enables a style of student/tutor interaction focused on the reasoning behind systems control behavior that better mimics proven Socratic human tutoring behaviors for highly cognitive skills. It supports fast, easy, and convenient authoring of such tutoring behaviors, allowing specification of detailed scenario-specific, but content-sensitive, high-quality tutor hints and feedback. The system places relatively light data-entry demands on the student to enable its rationale-centered discussions, and provides a support mechanism for fostering coherence in the student/tutor dialog by including focusing, sequencing, and utterance tuning mechanisms intended to better fit tutor hints and feedback into the ongoing context.

This work was done by Eric Domeshek, James Ong, and John Mohammed of Stottler Henke Associates, Inc. for Johnson Space Center. Further information is contained in a TSP (see page 1). MSC-24493-1

LEGEND, a LEO-to-GEO Environment Debris Model

LEGEND (LEO-to-GEO Environment Debris model) is a three-dimensional orbital debris evolutionary model that is capable of simulating the historical and future debris populations in the near-Earth environment. The historical component in LEGEND adopts a deterministic approach to mimic the known historical populations. Launched rocket bodies, spacecraft, and mission-related debris (rings, bolts, etc.) are added to the simulated environment. Known historical breakup events are reproduced, and fragments down to 1 mm in size are created.

The LEGEND future projection component adopts a Monte Carlo approach and uses an innovative pair-wise collision probability evaluation algorithm to simulate the future breakups and the growth of the debris populations. This algorithm is based on a new "random sampling in time" approach that preserves characteristics of the traditional approach and captures the rapidly changing nature of the orbital debris environment.

LEGEND is a Fortran 90-based numerical simulation program. It operates in a UNIX/Linux environment.

This work was done by Jer Chyi Liou and Doyle T. Hall of Johnson Space Center. Further information is contained in a TSP (see page 1). MSC-24805-1