Software for Optimizing Quality Assurance of Other Software

Software assurance is the planned and systematic set of activities that ensures that software processes and products conform to requirements, standards, and procedures. Examples of such activities are the following: code inspections, unit tests, design reviews, performance analyses, construction of traceability matrices, etc. In practice, software development projects have only limited resources (e.g., schedule, budget, and availability of personnel) to cover the entire development effort, of which assurance is but a part. Projects must therefore select judiciously from among the possible assurance activities. At its heart, this can be viewed as an optimization problem; namely, to determine the allocation of limited resources (time, money, and personnel) to minimize risk or, alternatively, to minimize the resources needed to reduce risk to an acceptable level. The end result of the work reported here is a means to optimize quality-assurance processes used in developing software. This is achieved by combining two prior programs in an innovative manner:

- *First Program*: The first of these programs is the Advanced Risk Reduction Tool (ARRT), which can be used to calculate the costs and benefits of a set of assurance activities on a given software project. ARRT is itself based on a risk-management tool, Defect Detection and Prevention (DDP). DDP uses a detailed mathematical model of requirements, risks, and mitigations.
- *Second Program*: The second of these programs is the TAR2 "treatment learner," which can be used to determine from a large set of factors those factor settings most critical to attaining a given objective.
- *Innovative Combination*: The major contribution of this work is the combination of these two programs. They are combined so as to operate in an iterative procedure, as follows: In each cycle of the iteration, TAR2 is tuned to identify the most critical software assurance activities, both those most critical to perform (because they contribute to cost-effective risk reduction), and those most critical to not perform (because they detract from cost-effective risk reduction).

These identified activities are then set accordingly in ARRT, and the cost-benefit calculations rerun. Repeating this cycle determines more and more activities to perform (and/or to not perform), culminating in a solution that is (near) optimal. An important aspect of this approach is that it allows for human experts to add further guidance during each iteration of the cycle. For example, if the experts observe that two of the recommended activities are actually incompatible (say, because they would both require use of the same limited resource at the same time), they can reject the TAR2 recommendations involving this pair of activities, and instead ask for the next-best solution. This makes good use of the experts' time, since they are only asked for guidance pertinent to promising solutions.

This innovation was developed by Martin Feather and Steven Cornford of Caltech and Tim Menzies of the University of British Columbia 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-30512.

The TechSat 21 Autonomous Sciencecraft Experiment

Software has been developed to perform a number of functions essential to autonomous operation in the Autonomous Sciencecraft Experiment (ASE), which is scheduled to be demonstrated aboard a constellation of three spacecraft, denoted TechSat 21, to be launched by the Air Force into orbit around the Earth in January 2006. A prior version of this software was reported in "Software for an Autonomous Constellation of Satellites" (NPO-30355), *NASA Tech Briefs*, Vol. 26, No. 11 (November 2002), page 44.

The software includes the following components:

- Algorithms to analyze image data, generate scientific data products, and detect conditions, features, and events of potential scientific interest;
- A program that uses component-based computational models of hardware to analyze anomalous situations and to generate novel command sequences, including (when possible) commands to repair components diagnosed as faulty;
- A robust-execution-management component that uses the Spacecraft Command Language (SCL) software to enable event-driven processing and low-level autonomy; and
- The Continuous Activity Scheduling, Planning, Execution, and Replanning (CASPER) program for replanning activities, including downlink sessions, on the basis of scientific observations performed during previous orbit cycles.

This program was written by Robert Sherwood, Russell Knight, Gregg Rabideau, Steve Chien, Daniel Tran, Benjamin Cichy, Rebecca Castaño, Timothy Stough, and Ashley Davies 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-30784.

Software for Analyzing Laminar-to-Turbulent Flow Transitions

Langley Stability and Transition Analysis Codes (LASTRAC) is a set of engineering software tools developed with the C++ language and modern software technologies for use in analyzing transition from laminar to turbulent flows. LASTRAC is a product of ongoing NASA Langley research projects related to transition flow physics modeling and simulations. It is intended to be a set of easy-to-use engineering tools that can be applied to routine engineering design studies. At the current stage, LASTRAC is capable of performing transition calculations based on linear stability theory (LST) or linear and nonlinear parabolized stability equations (PSE) for a broad range of flow regimes and configurations of interest for the design of low-speed as well as supersonic and hypersonic vehicles. At present, LASTRAC is limited to two-dimensional, axisymmetric, or infinite