Ensemble: an Architecture for Mission-Operations Software

Several issues are addressed by capitalizing on the Eclipse open-source software framework.

NASA's Jet Propulsion Laboratory, Pasadena, California

"Ensemble" is the name of an open architecture for, and a methodology for the development of, spacecraft mission-operations software. Ensemble is also potentially applicable to the development of non-spacecraft mission-operations-type software.

Ensemble capitalizes on the strengths of the open-source Eclipse software and its architecture to address several issues that have arisen repeatedly in the development of mission-operations software: Heretofore, mission-operations application programs have been developed in disparate programming environments and integrated during the final stages of development of missions. The programs have been poorly integrated, and it has been costly to develop, test, and deploy them. Users of each program have been forced to interact with several different graphical user interfaces (GUIs). Also, the strategy typically used in integrating the programs has yielded serial chains of operational software tools of such a nature that during use of a given tool, it has not been possible to gain access to the capabilities afforded by other tools. In contrast, the Ensemble approach offers a low-risk path towards tighter integration of mission-operations software tools.

Ensemble is based on an adaptation of the Eclipse Rich Client Platform (RCP), which is a widely used, readily available, stable, supported software framework for component-based development of application programs. The Eclipse RCP is a set of Java classes that define an architecture for general component-based application programs. New application programs are built on top of the RCP as a set of components, called plug-ins, that augment and extend its functionality. For example, a mission-activity-planning application program would consist of the RCP plus a set of plug-ins responsible for displaying, editing, and modeling activity plans. Application programs built on top of the RCP also gain access to a variety of such generally applicable capabilities as a help system, an update manager, and an extensible GUI.

In Ensemble, the difficulties of establishing interfaces between different software tools are minimized by developing most of the tools as Eclipse plug-ins. In addition, Ensemble draws upon capabilities provided by the Eclipse RCP to document and enforce interfaces between different components. In some cases, it may not be possible or prudent to develop a tool as an Eclipse Java plug-in. Such a tool can still be integrated with the Ensemble architecture. Development of a general, robust method of integrating non-Eclipse tools with other Ensemble tools is proceeding.

In Ensemble, the Eclipse framework provides a common GUI that can accommodate GUI components from multiple software tools developed by different teams. To the user, the resulting GUI looks as though it belongs to a single such tool while drawing on the resources of many of them. Ensemble provides for a task-oriented GUI that is based heavily upon an Eclipse perspective, which defines which GUI components are visible to a user at a particular time. As a user moves through tasks required for planning mission operations, the user clicks through a set of icons devoted to each task.

The combination of component-based development and a perspective-based GUI facilitates reuse of any software component at multiple stages of the operations process. In the past, a spacecraft-mission plan would be handed from one software tool to the next in a serial fashion. At each step, a single tool would exert exclusive control over the plan. In contrast, Ensemble plug-ins interact as a group with a common model of an evolving spacecraft plan. Each plug-in can contribute to the plan whenever it is necessary, and each plug-in must respond appropriately to modifications made by other plug-ins.

Most mission-operations-software development teams strive to make their software products applicable to multiple missions, but a typical mission does not need all the capabilities provided by a typical such product. To relieve a mission of the burden of maintaining, learning to use, and testing the software functions that it does not need, Eclipse provides for the distribution, to each mission, of only the core RCP plus only those plug-ins that afford the specific capabilities required by that mission.

This work was done by Jeffrey Norris, Mark Powell, Jason Fox, Kenneth Rabe, and I-Hsiang Shu of Caltech and Michael McCurdy and Alonso Vera of Ames Research Center 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 Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-41814.

Object Recognition Using Feature-and Color-Based Methods

The combination of methods works better than does either method alone.

NASA's Jet Propulsion Laboratory, Pasadena, California

An improved adaptive method of processing image data in an artificial neural network has been developed to enable automated, real-time recognition of possibly moving objects under changing (including suddenly changing) conditions of illumination and perspective. The method involves a combination of two prior object-recognition methods — one based on adaptive detection of shape features and one based on adaptive color segmentation — to enable recognition in situations in which either prior method by itself may be inadequate.

The chosen prior feature-based method is known as adaptive principal-component analysis (APCA); the chosen prior color-based method is known as adaptive color segmentation (ACOSE). These methods are made to interact with each other in a closed-loop system