







THE ENSEMBLE  
PROJECT ICON  
REPRESENTS THE  
COLLABORATIVE  
SPIRIT OF  
INDIVIDUAL  
ELEMENTS  
WORKING  
TOGETHER TO  
PROVIDE A MORE  
USEFUL WHOLE.

- **Owned** — Each Ensemble member will develop some capabilities that fall within an area that they consider their principal focus. The plugins that make up these capabilities are considered to be exclusively owned by the member and are clearly marked in the Ensemble repository with the reverse domain name of the team at the beginning, for example, *gov.nasa.arc.teamname.foo.bar*. The source code of these plugins is always available for reference by other Ensemble participants, but all modifications are submitted to the member that owns the plugin. These modifications are also considered the property of the member. In other words, contributions to an owned plugin do not grant the contributor any ownership of the plugin. Ensemble members may use an owned plugin in a project only with the permission of the member that owns it.
- **Shared** — Some capabilities (for example, parsing an activity dictionary) are utilitarian in nature and should be developed only once and maintained centrally. The plugins that make up these capabilities are developed cooperatively by Ensemble members and are marked in the Ensemble repository with names of the form *gov.nasa.ensemble.foo.bar*. All Ensemble members may modify and use these plugins for any purpose, regardless of whether they contributed any code to its development.

Owned plugins can become shared at the discretion of the member that owns them, but shared plugins cannot become owned without the approval of all Ensemble members. If a member decides to leave the Ensemble project, they retain the right to use all shared plugins according to the terms above and may remove all of their owned plugins from the Ensemble repository.

### Conclusion

Ensemble is enabling NASA missions to derive greater results from their investment in mission operations software. Instead of stringing together a series of largely isolated and independent tools, missions are free to assemble precisely the tools they need by drawing components together from different development teams. Mission operators become more efficient, which improves the overall performance of their missions. Finally, operations software developers are free to focus more on developing great tools and less on frustrating integration issues.

### Contact Information

The Ensemble Project is a collaborative effort of its member organizations. As currently constituted, the member Agencies and Centers are represented by their Ensemble Point of Contact:

#### Jet Propulsion Laboratory

California Institute of Technology  
4800 Oak Grove Drive, Pasadena, California 91109  
[www.jpl.nasa.gov](http://www.jpl.nasa.gov)  
Ensemble Point of Contact: David.S.Mittman@nasa.gov

#### NASA Ames Research Center

Moffett Field, California 94035  
[www.nasa.gov/centers/ames](http://www.nasa.gov/centers/ames)  
Ensemble Point of Contact: Michael.McCurdy@nasa.gov

#### NASA Johnson Space Center

2101 NASA Parkway, Houston, Texas 77058  
[www.nasa.gov/centers/johnson](http://www.nasa.gov/centers/johnson)  
Ensemble Point of Contact: Timothy.A.Hall@nasa.gov

This publication was prepared by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. © 2011. All rights reserved.

# The Ensemble Canon

## Introduction

Ensemble is an open architecture for the development, integration, and deployment of mission operations software. Fundamentally, it is an adaptation of the Eclipse Rich Client Platform (RCP), a widespread, stable, and supported framework for component-based application development. By capitalizing on the maturity and availability of the Eclipse RCP, Ensemble offers a low-risk, politically neutral path towards a tighter integration of operations tools.

The Ensemble project is a highly successful, ongoing collaboration among NASA Centers. Since 2004, the Ensemble project has supported the development of mission operations software for NASA's Exploration Systems, Science, and Space Operations Directorates.

## Background Information

Though Ensemble is based heavily upon Eclipse, a full description of this environment is well beyond the scope of this document. Numerous online and printed resources are available that describe Eclipse in great detail and should be considered a supplement to the brief description provided below.

Eclipse is most well known as an integrated development environment (IDE) for the Java programming language. Although Ensemble members often elect to use Eclipse as their IDE, Ensemble is not based on the Eclipse Java IDE but on the RCP. The Eclipse RCP is a set of Java classes that defines an architecture for general component-based applications. New applications are built on top of the RCP as a set of components called plugins that augment and extend its functionality. The Eclipse IDE for Java programming, for instance, consists of the RCP plus a set of plugins that add capabilities like compiling and debugging Java programs. A mission activity planning application would consist of the RCP plus a set of plugins responsible for visualizing, editing, and modeling activity plans. Applications built on top of the RCP also gain access to a variety of generally applicable capabilities such as a help system, update manager, and an extensible graphical user interface (GUI).

The Eclipse RCP is designed primarily as a framework for Java applications. Although it is possible to integrate components written in C or C++ using the Java Native Interface, this strategy is only feasible for components that don't have a GUI component. There is no straightforward way to tightly integrate a GUI written in a different language with the Eclipse GUI.

## Problem Statement

The current approach used to develop mission operations software has produced a set of powerful tools that have enabled stunning successes for NASA. Many parts of the current development process are functioning well and should be preserved. However, improvements in the state of the art in software engineering and increasing demands from new missions have exposed several areas that deserve attention. The five problems that Ensemble has been designed to address are outlined below.

### Brittle interfaces

Historically, mission operations software has consisted of a set of largely independent tools communicating with each other using files or socket-based interfaces. For example, the Mars Exploration Rover (MER) Activity Planning and Sequencing Subsystem (APSS) is made up of five tools that pass information among each other primarily using files and translators. The interfaces between the tools were added late in the development process and were the source of numerous problems. MER lessons learned workshops have identified these interfaces as an area that needs immediate improvement.

... IMPROVEMENTS  
IN THE STATE  
OF THE ART  
IN SOFTWARE  
ENGINEERING  
AND INCREASING  
DEMANDS FROM  
NEW MISSIONS  
HAVE EXPOSED  
SEVERAL AREAS  
THAT DESERVE  
ATTENTION.

### Too many GUIs

The number of separate tools used in the MER APSS is also the source of a popular complaint because it requires mission operators to interact with many different user interfaces to get their work done. This slows the overall pace of mission operations and increases training requirements.

### Difficult integrations

Development teams often need to build interfaces between their tools; a process that is often complicated by the fact that they have developed their tools in different environments using different standards. Teams have difficulty understanding the architecture of each other's tools, leading to poorly conceived integration plans.

### Duplication of effort

The tools used in existing mission operations systems are designed to address the needs of a certain phase of the operations process. One tool is designed to accomplish science planning while another is used for command sequencing. However, certain capabilities are needed at multiple stages in the operations process. Unfortunately, the architectures used in current mission operations tools do not allow capabilities from different tools to be reused at multiple steps in the process. As a result, redundant versions of these capabilities are developed by multiple teams and inserted into separate tools.

### Lack of agility

Most development teams strive to make their tools applicable to multiple missions. This is a positive goal because it enables future missions to capitalize on the investment made by prior missions. However, it can also force a mission to accept and maintain capabilities that it doesn't need. The popular "core and adaptation" model is an attempt to insulate different customers from customer-specific requirements, but what if one customer only needs a fraction of the core? Currently, that customer is simply forced to accept the rest of the core anyway, along with the risk and costs associated with its maintenance.

## The Solutions Offered by Ensemble

Ensemble addresses the five problems listed above by offering a superior environment for mission operations tools development. The details of Ensemble's approach to these problems are described below.

### Replace network and file interfaces with direct application interfaces

File and socket-based interfaces are notoriously difficult to test and debug. As a result, these kinds of interfaces tend to fail often. The most reliable interface between two tools is usually accomplished by direct use of the respective tools' application programming interface (API). This approach ensures that many problems in the interface are discovered at compile time.

Unfortunately, direct APIs are very difficult to implement when the tools being integrated were developed in different environments. Because most Ensemble members agree to develop their tools as Eclipse plugins, these issues are minimized. In addition, Ensemble draws upon capabilities provided by the Eclipse RCP (Extension Points, Plugin Dependencies, etc.) to document and enforce interfaces between different components.

In some cases it is not possible or prudent to develop a tool as an Eclipse Java plugin. These components can still be integrated with the Ensemble architecture. We have developed a general, robust method for non-Eclipse tools to interact with other Ensemble tools. By reusing a single interface point for multiple external tools, it is expected that the reliability of these interfaces will be increased.

### Provide a unified GUI for all operations tools

The complexity of mission operations makes it infeasible to develop a single operations tool capable of accomplishing all necessary tasks. However, Ensemble's reliance on Eclipse provides a common GUI framework that can contain GUI components from multiple tools developed by different teams. The result is a GUI that feels like a single tool to the user, but draws upon the resources of many development teams.

Ensemble uses a task-oriented GUI that is based heavily upon Eclipse perspectives. A perspective defines which GUI components will be visible to a user at a particular time. As users move through the tasks required for planning, they click through a set of icons devoted to each task. For instance, a user might click on an icon to make active a perspective devoted to browsing downlink data and then move on to a perspective devoted to building a set of activities for planning.

### Ease integration with a standardized development environment

Much of the difficulty met with in intertool integration occurs because development teams are using dissimilar development environments. Ensemble encourages its members to develop within the Eclipse IDE to minimize these differences.

The Ensemble project provides additional development infrastructure resources that further integrate efforts between teams. For instance, a common bug tracking system makes it straightforward to transfer tasks between teams, and a common code repository makes code integration much more straightforward. A continuous integration server builds and tests Ensemble-based products and provides continuous feedback to team members on the status of their product builds.

### Enable reuse of components throughout the operations process and between missions

The component-based development model and the perspective-based GUI that Ensemble inherits from Eclipse enables a mission to easily reuse any component at multiple stages of the operations process. For instance, a data view that was historically available only during the sequencing phase of operations can be displayed and used at any time if that view is developed as an Eclipse plugin.

This reuse is possible because of the manner in which Ensemble plugins deal with the spacecraft plan. In the past, the spacecraft plan has been handed from one tool to the next in a serial fashion. At each step, a single tool had exclusive control over the plan. In contrast, Ensemble plugins interact as a group with a common model of an evolving spacecraft plan. Each plugin can contribute to the plan whenever it is necessary, and each plugin must respond appropriately to modifications made by other plugins.

As a multimission architecture, Ensemble also supports extensive reuse of components among missions. The vast majority of Ensemble plugins are mission-independent, and mission-specific plugins are clearly identified. Ensemble-based mission operations tools were used in support of the Phoenix Mars Lander and the Mars Exploration Rovers, and are being developed for the Mars Science Laboratory mission as well as several technology programs. Each of these projects shares a large amount of source code.

### Support delivery of only those components that a customer requires

An Eclipse (and therefore Ensemble) application consists merely of the core RCP plus a set of plugins. The set of plugins that are included completely defines the functionality provided by the program. Because plugin dependencies are clearly documented and checked through the Eclipse IDE, it is possible to release only those plugins to a customer that provide the functionality they need. For instance, if a particular customer doesn't need spectral visualization support, the plugins that provide that capability can simply be omitted from their release. Plugin selectivity provides a tool that is easier for the customer to test and learn to use.

The plugin model provided by Eclipse enables Ensemble members to access a vast array of capabilities as needed while retaining the agility necessary to provide small, targeted releases that precisely meet a customer's needs. The Ensemble project's continuous build and test system has been designed to support this model.

## Guide for Ensemble Members

This section describes a few standards and practices that Ensemble members follow while working within the Ensemble project.

### Ownership

Inevitably, any project built on the concept of sharing must eventually address real issues of ownership. To reduce the possibility of misunderstandings in this area, Ensemble has adopted two classifications for plugins developed by Ensemble members:

... ENSEMBLE'S  
RELIANCE ON  
ECLIPSE PROVIDES  
A COMMON GUI  
FRAMEWORK ...

... THE ENSEMBLE  
PROJECT PROVIDES  
ADDITIONAL  
DEVELOPMENT  
INFRASTRUCTURE  
RESOURCES  
THAT FURTHER  
INTEGRATE EFFORTS  
BETWEEN TEAMS.