COMPONENT ARCHITECTURE
THE SOFTWARE ARCHITECTURE FOR MISSION REQUIREMENTS

Thomas Huang

Space Science Data Systems Section
Jet Propulsion Laboratory, NASA
Pasadena, CA 91109, USA
Thomas.Huang@jpl.nasa.gov

ABSTRACT

Software reuse is a common strategy in developing complex systems and has proven successful in reducing labor and maintenance costs. However, simply reusing modules will not produce a system that is adaptable to a variety of mission requirements. Because of this, projects often involve development of similar software systems from scratch in order to satisfy requirements. The end result is a system that can only operate in a specific environment and be used only in a specific way, with consequentially higher costs for maintenance and user training.

Component architecture consists of a framework that defines the standard interactions between components and standard interfaces for useful components to attach to the framework and interact with other components. Modern object-oriented programming languages are very good in their support for static interfaces, but need additional work in the area of dynamic interfaces. Reflection, which is available in some OO languages, should be considered in developing model component systems to enable dynamic discovery of service components at runtime. This enables software systems to be assembled at deployment time and provide users the ability to customize the software system with respect to their operating environment.

Our File Exchange Interface (FEI) is a file transaction service that offers portable, high performance, database-driven file management and transfer service. Unlike the common File Transfer Protocol (FTP), FEI provides file integrity verification on the fly, user authentication and authorization support, and database transaction management. FEI played a major role in file archiving and delivery service in flight missions such as Galileo, Mars Pathfinder, Deep Space 1, Cassini, and Space Infrared Telescope Facility. The new FEI version 5, code named Komodo, is a component-based service to enable pluggable support for various mission security requirements, database repositories, communication protocols, concurrency model, and file systems.

This paper presents the challenges in developing a dynamic service such as FEI to support various mission requirements while still being able to reduce cost on maintenance without sacrificing reliability and performance.

Keywords: Component, Component Configurator, Software Product Lines, Design Patterns, Reflection, Framework, Database.

1. INTRODUCTION

Despite dramatic increases in network and desktop computer performance, it remains difficult to design, implement, and reuse communication software for complex distributed systems. As the world's eyes and ears to the unknown frontier, the Multimission Image Processing System (MIPS) at JPL is expected to be able to accurately process all live science data gathered by spacecraft and distribute the processed data products to the science communities with respect to stringent quality-of-service (QoS) requirements. The image-processing framework, shown in Figure 1, consists of intelligent business components that perform acquisition and processing of telemetry data, cataloging of data products and onboard instrument states, visual verification and monitoring, science data processing, and distribution to subscribing science communities. While the framework defined the system's core services, each mission has its own set of requirements. These requirements may specify the method of telemetry data acquisition, visualization interface (if any), where and when data product distribution occurs, and most importantly of all the QoS