Astronomical Data Analysis Software and Systems IX
ASP Conference Series, Vol. 216, 2000
N. Manset, C. Veillet, and D. Crabtree, eds.

Multi-tier Internet Architecture with Java, UML and OOA & D

J. Chavez

Infrared Processing & Analysis Center/SIRTF Science Center, California Institute of Technology, M/S 314-6, 1200 E. California Blvd., Pasadena, CA, 91125

Abstract. The Space Infrared Telescope Facility[1] (SIRTF) is the fourth and final element in NASA's family of "Great Observatories" and is set for a December 2001 launch. SIRTF will provide infrared imaging and spectroscopy in the 3 to 180-micron wavelength range.

SIRTF will deploy an Internet Java application, the SIRTF Planning and Observation Tools (SPOT) to facilitate the proposal creation, editing and monitoring processes. Java will provide a development and runtime environment capable of supporting the diverse and globally dispersed astronomical community. The SPOT will provide a means of standardized input of observation parameters that can be integrated into the mission planning function using predefined workflow processes. The use of highly automated workflow processes will help keep the Development and Operations budgets within their current cost constraints.

The Science Operations Application Server (SOAS) is being designed and developed to support the SPOT and Science Operations System (SOS). SOAS will provide integration of back end services such as database, astronomical catalogs, and estimation software and legacy software libraries. Development of this application server architecture required the design and development of an object communications architecture and software deployment mechanism. The object communications architecture makes use of the built in serialization capabilities of Java and the Hypertext Transport Protocol (HTTP) for implementation. The software deployment mechanism is designed to provide initial software installation and an automated update facility and is implemented using a Commercial Off The Shelf (COTS) software package. Enterprise Java Beans will be developed and deployed within the SOAS to satisfy requests to the DBMS for proposal processing. This presentation will also cover the synergy between the UML and Java. The SOAS development team will utilize OOA&D, the UML and Java to design, document and implement the SOAS.

[1] `http://sirtf.caltech.edu`

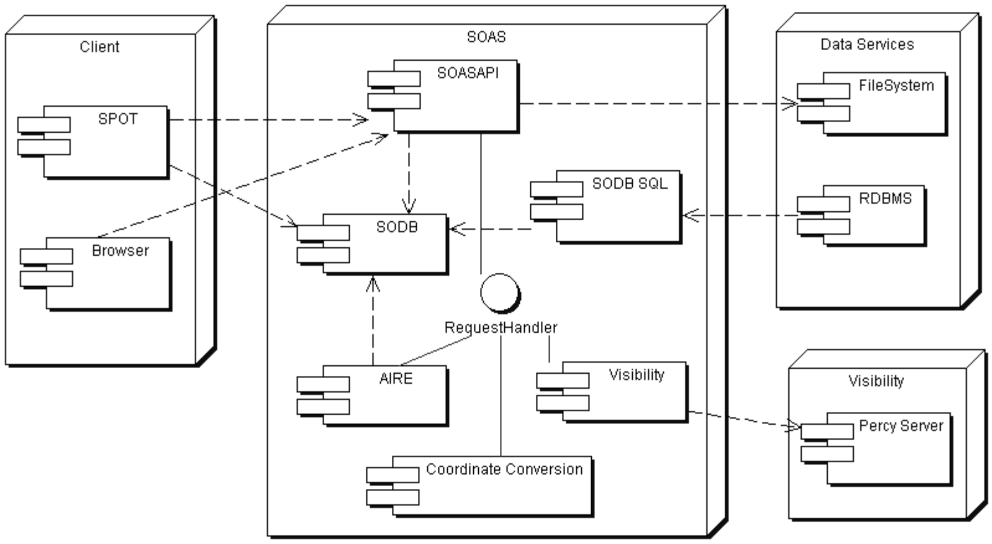

Figure 1. SOAS Deployment Diagram.

1. Introduction

The Science Operations Application Server (SOAS) supports the proposal creation, editing and submission activities the SIRTF Science Operations System. The architecture is a n-tier, high-availability, scalable, transactional and service based design. The design and analysis of the software architecture is based on Objected Oriented Analysis and Design techniques and is expressed in the Unified Modeling Language. The system is implemented using the Java Development Kit (JDK) and utilizes features from the Java 2 Enterprise Edition (J2EE). The SOAS API software architecture is comprised of services, data beans, exception handling and system monitoring components. The following sections describe the high level details of the SOAS.

2. Architecture

The SOAS is based on a n-tier architecture design with client (SPOT), application server (SOAS), and data (DBMS) tiers, see UML Deployment Diagram in Figure 1. The SOAS is comprised of the WebLogic[2] ^{TM} Application Server, Java 2 Enterprise Edition SDK and the Solaris operating system. All software runs on Sun hardware. The communications protocol between client and server is HTTP. Use of this protocol allows any Java or browser client with Internet access to communicate with the SOAS. Back end services (Data and Visibility) may select the most appropriate protocol suitable for integration with the SOAS.

[2] `http://www.weblogic.com/docs45/resources.html`

3. Design & Analysis

Applying OOA&D techniques on the high-level architecture reveals the set of SOAS components shown in Figure 1.

1. SOASAPI - The interface to the clients of the SOAS (SPOT and web browsers),
2. SODB - Science Operations Database object abstraction layer,
3. SODB SQL - Structured Query Language persistence layer to the RDBMS for SODB,
4. Request Handler - Interface definition for SOSAPI services,
5. AIRE Handler - Implementation of the Request Handler interface for AIRE services,
6. Coordinate Conversion Handler - Implementation of the Request Handler interface for Coordinate Conversion services,
7. Visibility Handler - Implementation of the Request Handler interface for visibility window calculations.

4. Implementation

The implementation of the SOASAPI is based on the Java Servlet API. The SOASAPI servlet is hosted by the WebLogic^{TM} Application Server. All clients communicate with the SOASAPI using a Uniform Resource Locator (URL) that corresponds to the requested service. There is a one-to-one mapping between URLs and SOASAPI services. The AIRE Visibility, SODB SQL components are implemented using the Java Enterprise Java Beans (EJB) API. The EJB components are hosted by the WebLogic^{TM} Application Server. Clients access the EJB components through the SOASAPI.

Services: Based on the URL the SOASAPI invokes the Request Handler to process that service. At initialization the available services are registered with the SOASAPI. The invocation of a particular service is encoded in the URL. Once the service has been determined the execution of the service is delegated to the appropriate Request Handler implementation.

SODB Data Beans: The SODB data beans encapsulate the database schema and SPOT ASCII file format. These data beans are also used to transfer data to/from the Java clients via Java object serialization. The SODB data beans support multiple version of the database schema through overrides of the default object serialization mechanism. This has been done to support backwards capability for the client ASCII file format. The SODB also contains a framework for intelligent of storage and retrieval of the data beans across the Internet. A partial SODB class hierarchy is shown in Figure 2.

System Monitoring & Control: Another set of services the SOASAPI provides are for system monitoring and control. A standard web browser is used to access these services. These services include:

1. Ping - check system availability,
2. Display Configuration - display the system configuration,
3. Reload Configuration - reconfigure the API dynamically.

Also provided by the WebLogic^{TM} Application Server is a system monitoring application.

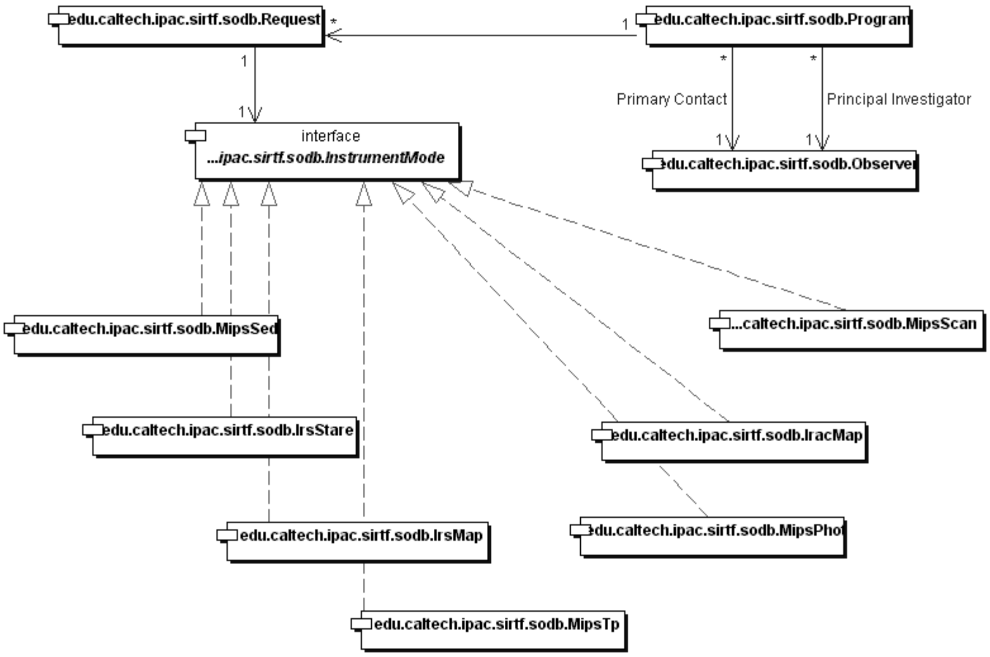

Figure 2. SODB Class Diagram (Partial).

Software Deployment: Initial deployment of the SPOT application is accomplished by the download and installation at the client workstation. From this point updates are processed using two methods. The first is a download of the complete updated application as mentioned. The alternative is to contact the SOAS for software updates. See Automated Software Update [3] pattern for more details on the second method.

References

Booch, G., Jacobson, I., & Rumbaugh, J. 1998, The Unified Modeling Language User Guide

Chavez, J. 1999, SDM/Uplink Integration Design Review

Harold, E. R. 1999, Java I/O

Hienrichsen, I. 2000, this volume, 319

Hunter, J., Craford, W., & Ferguson, P. 1998, Java Servlet Programming

Sun Microsystems 1999, Enterprise Java Bean Development Guide

[3] `http://www.joechavez.net/home/patterns/softwareupdate.html`