Software Architecture of the Spitzer Archive Interface

Joe Chavez, Xiuqin Wu, William Roby, Annie Hoac, Tatiana Goldina, Booth Hartley

California Institute Of Technology, Pasadena, CA, USA

Abstract. The Spitzer Science Center (SSC) provides a set of user tools to support search and retrieval of Spitzer Archive (SA) data via the Internet. This presentation describes the software architecture and design principles that support the Archive Interface subsystem of the SA (Handley 2007). The Archive Interface is an extension of the core components of the Uplink subsystem and provides a set web services to allow open access to the SA data set. Web services technology provides a basis for searching the archive and retrieving data products. The archive interface provides three modes of access: a rich client, a Web browser, and scripts (via Web services). The rich client allows the user to perform complex queries and submit requests for data that are asynchronously downloaded to the local workstation. Asynchronous download is a critical feature given the large volume of a typical data set (on the order of 40 GB). For basic queries and retrieval of data the Web browser interface is provided. For advanced users, scripting languages with web services capabilities (i.e. Perl) can used to query and download data from the SA. The archive interface subsystem is the primary means for searching and retrieving data from the SA and is critical to the success of the Spitzer Space Telescope.

1. Overview

The Spitzer Archive Interface (SAI) is a subsystem of the Spitzer Science Operation System (SOS) and is the primary means for accessing the data contained in the archive. The high-level requirements for the SAI are to provide for electronic-only download of archive data sets, facilitate large data sets, and provide flexible search and retrieval options to the user. Supporting the delivery of data solely via electronic means implies that the system will not generate hard media such as magnetic tape or compact discs. Our second high-level requirement, the ability to send the user a large data set (on the order of 40 GB) in an efficient manner, is paramount. Finally, the user will have the tools needed to select and download the desired archive data with a minimum of effort using supplied tools or via scripting languages. Each of the models is presented in UML 2.0 notation (Ambler 2005). Where appropriate the use of design patterns (Fowler 2002) is diagramed and discussed at a level that is appropriate for the scope of this paper.

2. Context

The SSA is a n-tier architecture, which contains at its core a typical client/server architecture. Figure II shows the SAI system context model, which for the pur-
pose of this paper consists of the Client, Services, Archive and Data Set nodes. The Client and Services nodes are the core of the SAI while the Archive database and Data Set file system are shared nodes. The Client node contains the software and configuration information to access the archive and other related (non-archive specific) services. The Services node contains the services to search the archive and retrieve archive data. The Downlink node represents the processing to create the Data Set files and the corresponding metadata in the Operations database. The Operations database is mirrored to the Archive database that contains a copy of the metadata set for the Data Set file system.

3. Use Cases

As part of requirements analysis, the use-case model (see Figure 2) was created to capture functional and non-functional requirements at a high level. There are a total of eight use cases defined in this model, only a few which can be described here. The two main use cases, as identified in the system requirements, are Search and Retrieve. The Retrieve use case includes the Stage use case that encompasses the system behavior that is required to package the requested data for download by the Archive User actor. A specialization of the Archive User actor, named Primary Investigator (shown as a directed arrow between the two actors), has the ability to log in to the system and request data that have not yet been made available to the Archive User.
4. Design Model

The Design model (shown in Figure 3) provides the details of the composition of the components of the SAI. The Client, Services and Data packages have detailed dependency relationships (shown as directed, dashed arrows). Each package is a set of compiled Java source code and supporting files. The Data package contains the SODB and Archive Common object model that together represent the data in the system. In the Client package the Leopard application provides the search and select functionality and the Subscriber application provides the staging and retrieve functionality. Both Leopard and Subscriber are based on the Spot Common package that provides common behaviors for GUI- and Internet-based applications. In the design model each of the services (Security, Search and Staging) are composed of client and server packages. Leopard depends on the Security package for user authentication and authorization, and on the Search package for searching the archive metadata set. Subscriber depends on the Staging Client package for the staging of data and the subsequent retrieval. There is a Web service interface and controller for each of these packages. For access to the Archive metadata, the Persistence package provides a framework to perform the object-to-relational mapping for the underlying RDBMS. The Staging Controller depends on the Packaging Service package which provides asynchronous staging of the requested dataset. This design allows for flexibility in the deployment of staging processes in a cluster configuration in order to handle a high volume of staging requests.
5. Deployment Model

The Deployment model, shown in Figure 4, was created to show the deployment of the physical components, as defined by the Logical and Design models. The diagram shows two physical execution-environment nodes: one for the client and one for the server. For each node, information is provided regarding the software stack and the operating system. Artifacts are packaged using standard J2EE packaging conventions. All the provided services use Simple Object Access Protocol (SOAP) layered on HTTP, except for the Auto Update service which is implemented as Web service and uses HTTP. Of note in the model is the deployment of the ArchiveRetrieveEJB.jar file onto the JBoss J2EE server cluster configuration.

References

Fowler, S. 2002, Patterns of Enterprise Application Architecture (Reading: Addison-Wesley)
Handley, T. 2007, in ASP Conf. Ser. 376, ADASS XVI, ed. R. A. Shaw, F. Hill, & D. J. Bell (San Francisco: ASP)