CS 578 – Software Architectures
Spring 2011
Homework Assignment #2
Due: *Tuesday, March 22, 2011*
– see course websites for submission details –

Introduction

In this assignment, you have to assess the designed and implemented architecture of Apache OODT (Object-Oriented Data Technology). OODT is the grid middleware that the Big Data Filesystem is built from in the first assignment.

Apache OODT is a data and compute grid technology designed for search and discovery of large-scale data that may be distributed across data nodes and the creation and management of workflows. OODT has been used to manage large scale distributed data applications in a variety of domains including space and planetary science, biomedical science, earth science and others. OODT provides a variety of large-scale data management services including data curation, crawling for data, workflow management, data and metadata servers and clients, etc.

OODT originated from the NASA Jet Propulsion Laboratory and has been in development for over a decade and has involved dozens of developers and several architects. The source code is largely built using Java, however, a Python client of OODT is available called Agile OODT. OODT currently includes 100KSLOC in primarily Java with 1000+ classes.

Different aspects of OODT's architecture are documented in multiple places, including the components that should exist, the connectors that they may use, the possible configurations of the components and connectors that make sense, and other design decisions:

- C. Mattmann, D. Crichton, N. Medvidovic and S. Hughes. A Software Architecture-Based Framework for Highly Distributed and Data Intensive Scientific Applications. In *Proceedings of the 28th International Conference on Software Engineering (ICSE06)*, pp. 721-730, Shanghai, China, May 20th-28th, 2006. http://sunset.usc.edu/~mattmann/pubs/ICSE06.pdf
- C. Mattmann, D. Freeborn, D. Crichton, B. Foster, A. Hart, D. Woollard, S. Hardman, P. Ramirez, S. Kelly, A. Y. Chang, C. E. Miller. A Reusable Process Control System Framework for the Orbiting Carbon Observatory and NPP Sounder PEATE missions. *In Proceedings of the 3rd IEEE Intl' Conference on Space Mission Challenges for Information Technology (SMC-IT 2009)*, pp. 165-172, July 19 - 23, 2009. http://sunset.usc.edu/~mattmann/pubs/SMCIT09.pdf

These are the papers that were referenced in your first assignment. Other useful information can be found from the following websites:

- Apache OODT website: http://oodt.apache.org/
- OODT from JPL's website: http://oodt.jpl.nasa.gov/oodt-site/
  The latter website includes many papers written about OODT and extra documentation about the different components and connectors of OODT. This documentation also includes examples of how to use OODT, developer documentation that includes UML diagrams of individual component's internal structure and behavior, and presentations given about OODT.

  The OODT components that are of interest for this assignment are listed below in two categories.

  Information Integration Components
  -----> Web Grid
  -----> Product Servers
  -----> Profile Servers
  -----> XMLPS
  -----> OPeNDAP PS

  Catalog and Archive Service Components
  ------>File Manager
  ------>Workflow Manager
  ------>Resource Manager
  ------>Crawler
  ------>PushPull
  ------>CAS PGE
  ------>PCS
  ------>Metadata

  These components are categorized below according to their size.

- **Very Large Components:**
  - File Manager
- **Large Components:**
  - PushPull, Workflow Manager, Resource Manager
- **Medium-sized Components:**
  - Metadata, Profile Server, XMLPS, Product Server, Crawler
- **Small Components:**
  - PCS, CAS PGE, Web Grid, OPeNDAP PS

  You must select a set of components to recover based on their size (and thus, indirectly, complexity). We ask that you select your components based on the following criteria:

  - **2 large components, 1 medium-sized component, and 1 small component; or**
  - **1 very large component, 1 medium-sized component, and 1 small component**

  For example, you can select this set of components: File Manager, Profile Server, and Web Grid.

As another example, you can select this set of components: PushPull, Workflow Manager, Crawler, and PCS.

Architectural Conformance

For each component that you select from above, choose an architectural decision given in the documentation of OODT (publications, presentations, and web pages). Architectural design decisions can be the components and connectors of OODT; the set of services provided by a component; interactions between components; or decisions that address non-functional properties, such as heterogeneity, modularity, or scalability.

For each decision, you will have to write a subsection. This subsection should describe:

1. The source of the architectural decision – from which piece of documentation (publication, presentation, or web page) does the decision originate
2. Where and how the decision is implemented in the source code
3. Whether the implementation conforms to the documented description. In cases where it does not conform, describe how the architectural decision is violated and discuss how the implementation could be repaired.

Use diagrams or other models to clarify your descriptions.

To aid you in this task, it is useful to obtain the dependencies between source code elements, particularly for determining the static configuration of components and connectors. We **require you** to use the following tool to extract dependencies between source-level elements of OODT:

- Class Dependency Analyzer (CDA): http://www.dependency-analyzer.org/. This is a Java dependency analysis tool that finds dependencies at the package and class levels. Features of CDA include query mechanisms, a GUI, and graph construction and visualization.

By requiring you to use this tool, we ask that any source level diagrams you produce be either based on dependencies from this tool or drawn based on dependencies between classes or packages analyzed using this tool. Requiring you to use this tool also allows us to expect the same dependencies to be found for the same set of classes and packages.
If you feel that you really need to examine the method-level dependencies between or within classes, you'll either need to read the source code or find a different tool. However, since the technique we ask you to use below works at the object-oriented class level and allows you to look at only a partial architecture, then examining the source code and methods should be minimized.

To identify components, you will use the architectural recovery technique, Focus:

- N. Medvidovic and V. Jakobac. Using software evolution to focus architectural recovery. *Automated Software Engineering Journal*, 13(2):225–256, 2006.

Focus is used to recover a partial architecture of the software system by targeting only the parts of the system affected by a change to requirements. Focus recovers the architecture of a software system using the architectural style most appropriate for the system. While Focus recovers the data components and processing components, it does not explicitly recover the connectors.

To identify connectors, you will need to search for names such as "socket," "stream," "opendap," etc. You can do this either manually or by using a query mechanism over the source code (e.g., a combination of the unix find and grep command-line programs or a search in Eclipse). From that point, you can manually look at the code to see which components may be connected to each other over these connectors.

Deliverable

The deliverable of your assignment is a written assessment report. The report should be written using the font Times New Roman in font size 12 and single line spacing for each paragraph. Please check your spelling and use proper grammar.

Please adhere to the following format when you answer your questions about the architectural decisions.

1. Name of component(s) involved in decision
    a. Source of Architectural Decision: (URL with title or Citation)
2. Explanation of decision goes here. You can include diagrams (e.g., from the publications, presentations or web pages)
    a. Where and how it is implemented?
3. Place explanation of implementation here. You can include diagrams (e.g., screenshots from CDA or component and connector diagrams like in the styles slide from lecture or in the textbook)
    a. Does the decision conform? How or how not does it conform?
4. Place explanation of conformance here. You can include diagrams.

Suggested assignment length (3-6 pages)

The length suggestion is only a guideline and can be extended as needed.