CS 578 – Software Architectures
Fall 2013
Homework Assignment #2

Due: *Monday, October 28, 2013*

– see course websites for submission details –

Background

In the process of a large and complex software-intensive system development, there are many design decisions that have to be made by software architects. The decisions range from big decisions such as picking the architectural style of the overall system or selection of off-the-shelf systems to small decisions such as choosing data type for a variable or defining the maximum size of an array. Those decisions are trade-offs; it is often the case that software architects have to choose among several options that have both pros and cons. Software architects have to be aware of the consequences of the options to make the best decision.

However, being aware of the consequences of design decisions is not easy, especially for large and complex software systems. Such big systems have large number of components and connectors that often depend on each other. One small change made to a part of the system could affect not just the component in which the change was made but also another component that relies on that component, and furthermore, there could even be other components that are also affected in sequence. The system could perhaps be multi-threaded, making it even harder to predict how the system would behave at runtime.

Model-driven engineering approaches such as the eXtensible Tool-chain for Evaluation of Architectural Models (XTEAM: http://softarch.usc.edu/~gedwards/xteam.html) for software systems have risen to provide the rationale that software architects need upon making design decisions. XTEAM enables both static and dynamic analyses of software models by supporting customization of architecture description language (ADL, software modeling notation in other words) and interpretation of the models.

Overview

The main purpose of this assignment is to give you the initial exposure to XTEAM. This assignment is the first of the three-step assignment design that, along the way, will expose you to the state-of-the-art software modeling technique.

In this assignment, you will perform trade-off analyses that shed light on a set of candidate architectural design decisions. You will modify, create, and analyze an architectural model of NGCA, specifically the part that handles sea salinity data as given in the NGCA description document for the last assignment. You will be provided with a pre-made ADL and a partly completed model so that you could start from there.

A separate document will be provided to guide you to download and setup XTEAM. You can get Visual Studio 2008 from DreamSpark for free, and the rest of the required software is either free or free for academic purposes.

The NGCA Model

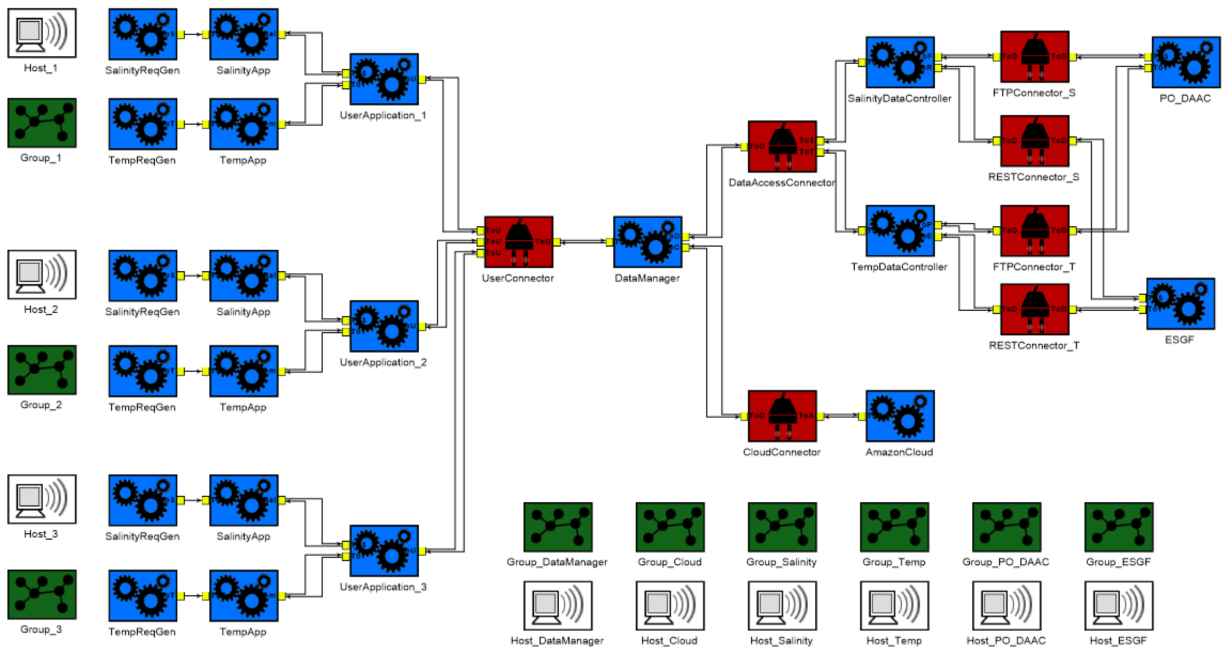

Figure 1: The NGCA model for this assignment

Figure 1 is a screenshot of the model that will be given to you for this assignment. It represents a part of the NGCA level 0 architecture described in the last assignment. Blue boxes represent components, and red boxes represent connectors. In addition to the example in the NGCA description document, sea temperature data has been added for this assignment.

Recalling the architecture of NGCA (depicted in Figure 2 of the NGCA description document), below list describes into which layer the components and connectors fall and how they behave.

Application Service Layer

Included elements:

- SalinityReqGen
- SalinityApp
- TempReqGen
- TempApp
- UserApplication
- UserConnector

Description:

SalinityReqGen and TempReqGen generate a data request repeatedly at each given time units. The requests are subsequently sent to the corresponding Apps (SalinityApp and TempApp), then to UserApplication, and eventually to UserConnector that forwards the requests to DataManager. Conversely, if UserConnector receives a response from DataManager, it forwards the response to the corresponding SalinityApp or the TempApp through the UserApplication that has requested the data earlier.

Data Management Layer

Included elements:

- DataManager
- DataAccessConnector
- CloudConnector

Description:

The DataManager component is responsible for retrieval of data from from the layer below (Data Access Layer) and the transformation of the format of the data. When it receives a request from a UserApplication, according to the kind of the data of the request (either salinity or temperature), it forwards the request to the corresponding data controller. When it receives a response from a data controller, it performs the transformation by allocating transformation jobs to the cloud computing infrastructure. A transformation job is created at the DataManager component and sent to AmazonCloud through the CloudConnector.

Data Access Layer

Included elements:

- SalinityDataController
- TempDataController
- FTPConnector
- RESTConnector
- PO_DAAC
- ESGF

Description:

The two data controller components retrieve data from the data sources, i.e. PO.DAAC and ESGF, through the corresponding interfaces, i.e. FTP and REST, when they receive data access requests from the Data Management Layer. The data sources generate data access responses and return them back to the components that requested the data.

Shared Infrastructure Support

Included elements:

- AmazonCloud

Description:

Shared Infrastructure Support is supposed to provide the cloud computation capability for transformation of data response. When AmazonCloud receives a data transformation request (in the form of a data response originated from the Data Access Layer), it performs the transformation, and returns the transformed data back.

Tasks

Use NGCA.mga as given for each task under the [distribution_package]\assortedxadl_model\ directory. Do not reuse a model with changes for one task for another task.

Select **two** tasks from the following list and perform them:

1. Suppose we have a non-functional property system requirement that no component/connector should consume more than 700,000 units of energy. Find all components and connectors that violate the requirement, and suggest possible changes that would satisfy the requirement.
2. Vary the initial size(+1 , -1) of DataAccessRequest and find out which properties (message latency, energy consumption, and memory usage) of the system are affected by that change. Report the results of the simulation before and after the change regarding the affected properties.
3. By further increasing the execution times of certain tasks in the behaviors of components that act as a “server” such as PO_DAAC, ESGF, and AmazonCloud, determine the point at which the system cannot handle its envisioned (initially modeled) capacity. Report what changes you made and how those changes resulted in detail.
4. Compare two different deployment: (1) a deployment that TempDataController and SalinityDataController are deployed on the same machine, and (2) a deployment that these two components are deployed on two different machines. Which deployment among the two is better/worse/same in terms of the three properties (message latency, energy consumption, and memory usage) and how?

For each of the two tasks that you selected, you should provide an updated or changed model based on task’s request and the output(s) of XTEAM with the model. You should provide discussions for those tasks that require you to report something or compare certain options in a PDF file. The name of the PDF file should be <lastname>_<firstname>_csci578_mattmann_HW2.pdf . All of the materials should be submitted in a zip file with the name <lastname>_<firstname>_csci578_mattmann_HW2.zip

Questions

Should you have further questions, send an email to both Reza Safi (gsafi@usc.edu) and Jae young Bang (jaeyounb@usc.edu)