CSCI 578 – Software Architectures

Fall 2013
Course Project

Due: *December 6, 2013, 12pm PT*

During the course of the semester, you have been introduced to a variety of canonical software architecture terminology, techniques, and technologies -- five of which we will directly weave together through this course project: *software architectural styles*, *domain-specific software architectures*, *connectors*, *middleware technologies* and *architectural recovery*.

We will assert that *the* key observation in the software engineering of scientific systems[1] in recent years is that such systems are being constructed and *used* in highly distributed environments – scientists across the world are working together with their colleagues in search of answers to previously unimaginable scientific problems (e.g., accurate and precise early detection of cancer, mapping out of the human genome, earthquake simulation, high energy physics computations, and the like). The principal enabling technology of these systems has been *grid-computing* technologies. Grid computing connects dynamic collections of individuals, institutions, and resources to create virtual organizations which support sharing, discovery, transformation, and distribution of data and computational resources. Distributed workflow, massive parallel computation, and knowledge discovery are only some of the applications of the grid.

In the last few years, our research group at USC has studied a number of grid computing technologies by examining their *as*-implemented architectures, and comparing and contrasting them with their *as*-intended architecture, a five-layer grid "reference architecture" by Kesselman et al.[2], shown in Figure 1. We published the results of one such study[3] in 2005, and an addendum to the study in 2009[4]. In these studies, we examined eighteen off-the-shelf, open source grid-computing technologies, including a major data-grid technology called OODT, developed by NASA, and *the* pervasive computational grid technology, Globus. The results of our study yielded three critical conclusions: (1) the requirements for grid systems are very broad, and because of this, it is hard to discern the exact intention of the grid requirements over those of traditional middleware; (2) there is overlap between the five layers of the grid reference architecture,

---

[1] Software systems that support scientists in search of observation, discovery, and the collection, management and distribution of massive amounts of *data*

[2] C. Kesselman et al. The Anatomy of the Grid: Enabling Scalable Virtual Organizations Intl J. Supercomputer Applications, 2001.

[3] C. Mattmann et al. Unlocking the Grid. In Proc. of Component-Based Software Engineering (CBSE), 2005.

[4] C. Mattmann et al. The Anatomy and Physiology of the Grid Revisited. In Proc. of the WICSA/ECSA, 2009.

as evidenced by the fact that several as-implemented components share functionality across more than one layer; and perhaps the most critical conclusion, (3) *grid technologies regularly violate the grid reference architecture*. Of the eighteen we studied all violated the grid reference architecture in various forms: (1) upcalls – calls made from components in a below ("servicing") layer making calls to components in an above ("client") layer, (2) crossing of 2 layer boundaries – components in one layer making calls to other components, either above or below, at a distance of > 1 layer, and (3) dependencies between layers that were not specified in the reference architecture.

Given this knowledge, and these somewhat frightening conclusions, in our research groups at USC and JPL we have made it an emphasis to thoroughly think through the implications of architectural decisions in the development and analysis of grid software architectures – to our betterment. In addition, several of the developers of modern grid technologies appear to have (at least anecdotally) understood, and corrected, some of the problems in the technologies that we examined back in 2004-05 for our original study, and again in 2009 for our follow-on study.
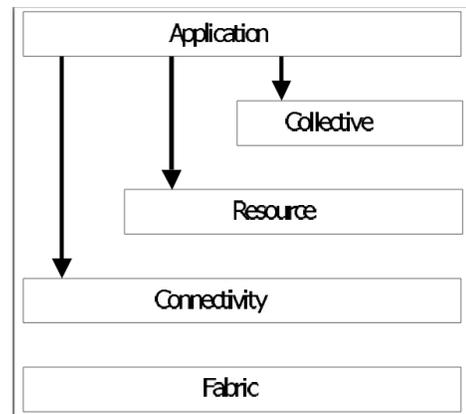

Figure 1. Grid Reference Architecture: Fig. 2 from Kesselman et al.

However, the question that we have been batting around in recent months, and the question that you and your classmates will help to answer through your course project is: *how much* have the developers of grid software systems learned in the last decade with respect to matching architectural intention to as-implemented needs? In other words, how many of the identified discrepancies with grid technologies still exist in a representative subset of them? Furthermore, are there any new discrepancies that have emerged, showing *architectural drift* between the grid "reference architecture" and the as-implemented architectures of modern grid technologies? You and your classmates will examine these questions in the context of the Next Generation Climate Architecture (NGCA) case study that we have used throughout the course. Indeed, the NGCA architecture and technologies that implement it *can be classified as grid* technologies – and we have established collaborations with teams working on NGCA systems to provide their code to you and your classmates to investigate and to recover their as-implemented architectures!

Project Description

Outline:
1. Group Formulation
2. Selection of Architectural Recovery Technique
3. Assignment of NGCA technology
4. Architectural Recovery
    a. Components

      b. Connectors
      c. Styles
5. Shoe-horning of recovered architectural elements into NGCA architecture
6. Discrepancy identification
      a. Upcalls
      b. 2 Layer boundaries
      c. Unspecified layer dependencies
      d. Other, unidentified discrepancies
7. Deliverables
      a. Project Report
      b. Analysis Data
          i. Incremental, step-by-step diagrams of architectural recovery steps
          ii. Final recovered architectures of studied NGCA technology
          iii. Shoe-horning diagrams of components into reference architectures

---

1. Group Formulation (due date: 11/18/2013)

You are required to form groups of between 2 to 3 students. On-campus students are responsible for coming to class during the week of November 18th, and forming your groups. If you are in the DEN section of the class, or if you are having difficulty forming a group, please contact the TAs who will assist in helping to formulate groups.

Once you have successfully agreed on a group, please have 1 person from the group email the names of the people in the group, along with their email address to the TAs.

2. Selection of Architectural Recovery Technique (due date: 11/19/2013)

Since you will be studying the as-implemented architectures of NGCA technologies, you will need to select an appropriate architectural recovery technique to employ that takes the as-implemented software and results in the recovery of (at least) a partial architecture[5] that you can use for the rest of the project. We have selected three main architectural recovery techniques. All groups will use *Focus*[6] as the over-arching recovery technique, since it is able to capture architectural information (such as connectors) that other techniques neglect to identify. You will then choose **one** of the other two techniques below besides Focus (either Rigi or PBS) to perform semantic clustering of code into architectural components. We have provided links to appropriate documentation on all of the techniques in the table below. You will need to carefully review the documentation on each technique to get a feel for how to use the one that you select.

Technique	Pros	Cons	Link
Rigi	Tool Support, very expressive	Moderate-high complexity	http://www.rigi.csc.uvic.ca/
Portable	Tool Support	Moderate	http://www.program-

---

[5] We say "partial" because the technique may not (and most likely *will not*) yield a full architectural specification. It will probably suggest appropriate components, information about their dependencies, and perhaps connectors.

[6] http://sunset.usc.edu/~neno/Focus/

		complexity	transformation.org/Transform/DaliW orkbench
Book Shelf (PBS)			

3. Assignment of NGCA Technologies (due date: 11/22/2013)

In our above-referenced CBSE paper[1], we focused on five grid technologies: OODT, GLIDE, Globus, DSpace and JCGrid. For your course project, you will be assigned **one** NGCA technology as soon as your team has been formed (and approved by the TAs), and as soon as your recovery technique has been selected (and approved by the TAs):

Topical NGCA Technologies

Technology	Link
CDS Client (NASA MERRA Analytics Services)	http://www.nas.nasa.gov/SC12/demos/ demo24.html
CMAC Carnegie Mellon Workflow Recommender	http://repository.cmu.edu/cgi/viewcont ent.cgi?article=1118&context=silicon _valley
Apache Open Climate Workbench (OCW)	http://climate.incubator.apache.org/

Note that all of the above NGCA software systems should be freely available and you either already have access to their source code (in the case of Apache OCW), or it will be provided to you. If you cannot find the source code or if it has not been provided to you by 11/22/2013, please contact the instructor or the TAs for help. The TAs will notify your group of its assigned NGCA technologies by November 22, 2013.

4. Architectural Recovery

Now that you have selected both your recovery technique, and been assigned your NGCA technology, your mission is to perform architectural recovery on its implementation. Use the documentation for the architectural recovery technique chosen, along with any tool support (either provided, or that you develop), or manual effort, with the following recovery goals:

- **Recovery of the Major Software Components in each NGCA Technology**– As you've probably guessed, there is no magic number of components that belong to any one of these NGCA software technologies. The appropriate number of components is an answer that you will need to arrive at through careful research, diligence and plain old-fashioned luck. Make sure that you can justify your groupings of implementation code into functional components. Check the NGCA technology documentation – search for a reference architecture, look online, etc. In some ways, you will be constrained by the explicit support for component identification provided by the group's selected architectural recovery technique. That is, if you choose Rigi, your component recovery step will focus mainly on static analysis and actual code artifacts. You should make sure to carefully review the steps that each architectural recovery approach allows for as you recover the

components of the grid technology, and make sure to document your rationale.

- **Recovery of the Major Software Connectors in each NGCA Technology** – This step is going to be a bit trickier and even though you are explicitly using Focus as the overarching architectural recovery technique, you may need to develop your own means of identifying the appropriate connectors between the identified components in the NGCA technology that you are studying. Use the knowledge gained during the course on connectors, e.g., your study of connectors, as well as the course material/lectures/etc. and any ideas that you can come up with by examining documentation about the NGCA technology or online sources to identify the appropriate connector relationships between your software components that you recover. Try to be as thorough as possible, and document your rationale.

- **Recovery of Two Architectural Styles used in the NGCA technology** – Identify two major architectural styles (from the set discussed in class, e.g., P2P, Client/Server, etc.) used in the NGCA technology besides the layered architecture present in all grid systems and besides the initial NGCA architecture description. Justify your answers with appropriate rationale. Execute the NGCA technology if necessary, to gain more information, and use any information available to you: e.g., documentation, web sites, discussion boards, etc., to aid in your style identification.

5. Shoe-horning of recovered architectural elements into the NGCA architecture

Similar to the approach discussed our CBSE paper[3], once the architecture of the NGCA software system has been recovered you will "shoehorn" the architectural components (and their interconnections) into the NGCA reference architecture (shown in Figure 2).

To shoehorn the components into the layers, use any of the following information:
1. The component's recovered relationships (connectors) with other components
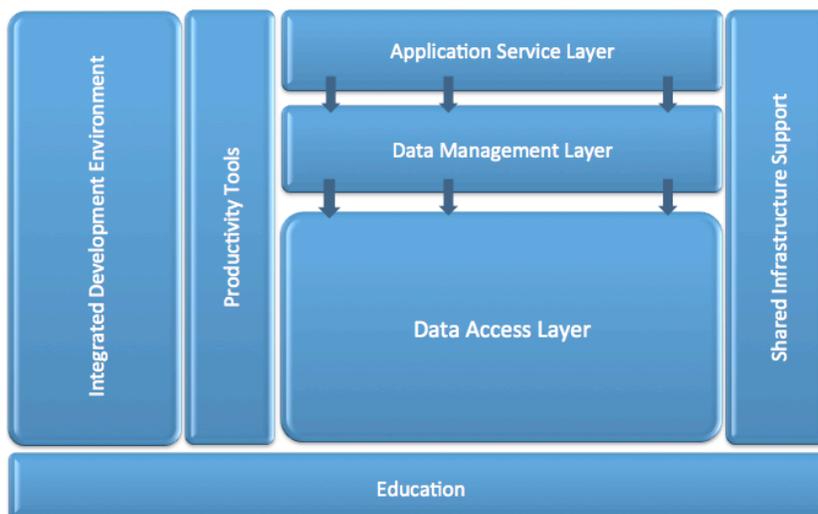

Figure 2. Next Generation Climate Architecture (repeated from HW #1)

2. The source code
   a. The package structure of the classes belonging to the component
   b. The component names
   c. The connector names (if explicit)
3. Documentation about the NGCA technology vis-à-vis documentation about the particular layer.
4. Any other reasonable technique that you can derive
   a. If you derive a new "shoehorn" technique, please document it with appropriate rationale.

Your group should produce diagrams bearing resemblance to Figs. 3 and 5 from our CBSE paper[3] as a result of this step.

6. Discrepancy Identification

In this step, you and your group will take the resultant "shoehorned" diagrams and use them to identify discrepancies in your assigned NGCA technology. You are responsible for identifying:

- **Upcalls** – As an example, consider a two layered software system, containing component B in layer 1 (the bottom layer) and component A in layer 2 (the top layer). An *upcall* is defined as the situation in which component B has an explicit dependency relationship (or connector/connection) to A: that is, B requires some service from A, or its presence in order to function as revealed by your architectural recovery and shoehorning process.
- **Crossing of 2 Layer Boundaries** – Consider a three-layered system in which there are layers 1 (the bottom layer), 2 (the middle layer) and 3 (the top layer). Consider components A in layer 3, and B in layer 1. A *crossing of 2 layer boundaries* is defined as the situation in which component A in layer 3 has a dependency relationship (or connector/connection) to B: that is A requires some service from B, or its presence in order to function as revealed by your architectural recovery and shoehorning process. This is also true in the case where B relies on A.
- **Unspecified layer dependencies** – Consider a three-layered system in which there are layers 1 (the bottom layer), 2 (the middle layer) and 3 (the top layer). Consider components A in layer 3, C in layer 2, and B in layer 1. Consider also that the grid reference architecture shows a dependency between layers 2 and 1 and between layers 3 and 2. An *unspecified layer dependency* is defined as the situation in which e.g., component A depends on component B (i.e., crossing of a 2 layer boundary as described above), or vice –versa, B depends on A, indicating that in fact layer 3 depends on layer 1, or vice versa. Once you have determined the crossing of the 2 layer boundaries, this step should be relatively straightforward.
- **Other unspecified discrepancies** – This would include components that you couldn't "shoehorn" into a particular layer, or any other weird occurrences that you run into.

7. Deliverables (due date: 12/6/2013)

Once your group has completed steps 1-6, you are responsible for producing the following set of deliverables:

- **Project Report** – Create a project report of between 3-4 pages describing the following
    - o Brief Introduction and Motivation
    - o Architectural technique chosen and why
    - o NGCA assigned and brief description of it
        - ▪ What, in particular, are the purported benefits of the NGCA technology?
    - o Recovery Process
        - ▪ Rationale for component recovery and identification
        - ▪ Rationale for connector recovery and identification
        - ▪ Rationale for architectural style recovery and identification
    - o Steps that were difficult about the assignment and why
    - o Steps that were not as difficult about the assignment and why
    - o Conclusions
        - ▪ Your group's specific findings, summarized
        - ▪ Do you agree with the three major conclusions from the CBSE study, based on your course project? Why, or why not?
- **Analysis Data** – provide the following data that you collected during your project
    - o *Incremental, step-by-step diagrams* of the architectural recovery steps that took you from source code to components, connectors, and styles. These diagrams should be in one of the accepted course diagram formats, so check the homework guidelines for more details. You should also provide, or point us to the original source code that you performed architectural recovery on (give us a URL to the source tar ball, or zip file, preferably, or if the source code is < 10 MB, include it in your submission).
    - o *Final recovered architectures* of the NGCA technology that you studied, in the form of one "final" architectural diagram. Again, consult the homework guidelines for acceptable diagram formats.
    - o *Shoehorning diagrams*, one for your NGCA technology, showing your "shoehorning" of the components and connectors into the NGCA reference architecture.

You will submit your homework deliverables according to the homework submission guidelines. You should use an easy-to-recognize project directory structure, something like the following:

- `./<groupXXX>_projectreport.doc`
- `./analysis`
- `./analysis/step-by-step`
- `./analysis/step-by-step/<NGCA technology>/`
- `./analysis/step-by-step/<NGCA technology>/fig1.xxx`
- `./analysis/final`

- `./analysis/final/<NGCA technology>`
- `./analysis/final/<NGCA technology>/final-arch-fig.xxx`
- `./analysis/shoehorn`
- `./analysis/shoehorn/<NGCA technology>`
- `./analysis/shoehorn/<NGCA    technology>/final-shoehorn-fig.xxx`

Once created, zip the root level directory for your project up according to the homework submission guidelines and submit the project as you would any of the other homework assignments.

If you have any further questions or concerns, please do not hesitate to contact the TAs or the Instructor. Good luck!