Homework 2

The purpose of this homework is perform the detailed planning and design required for the Lifecycle Architecture milestone.

The results of this work will be presented as the Life Cycle Architecture milestone (LCA). Refer to the lectures on the LCA review and architectural description, as well as the various reference papers and web pages, for background material on the content of this review.

The homework is due before midnight, (*revised*) **Monday May 3**. One of the team members should turn in all the deliverables. We will schedule 20-minute sessions throughout the day on **Tuesday May 4** for LCA reviews with each team.

Deliverables

Much of the material for this review is an elaboration of the material that was written for the LCO reviews. Feel free to draw on that as a starting point. However, remember that you are defining the detailed architecture now, and so there should be fewer options and open items, and more decisions and detail.

1. An overview presentation. A set of Powerpoint presentation slides that summarizes the LCA elements for your product.

2. Specification document. This document should accurately reflect the product you are building, from the point of view of both the client user and the product administrator. See the lectures and the below referenced Spolsky article for suggestions on content.

3. Architecture document. Detailed definition of the system and software components. See the lectures and the attached extract from *Software Architecture* by Garlan for suggestions on content.

4. Schedule and task assignments. Milestones, task descriptions, and the specific team member responsible for each task. This should reflect your actual plan of work.

References

Anchoring the Software Process, Barry Boehm, USC
http://citeseer.nj.nec.com/boehm95anchoring.html

Painless Functional Specifications, Joel Spolsky
http://www.joelonsoftware.com/articles/fog0000000036.html

Software Architecture, David Garlan
http://www-2.cs.cmu.edu/afs/cs/project/able/ftp/encycSE2001/encyclopedia-dist.pdf

Architectural Description Languages. Extracted from Garlan.

The first insight is that good architectural description benefits from multiple views, each view capturing some aspect of the system. Two of the more important classes of view are:

- Code-oriented views, which describe how the software is organized into modules, and what kinds of implementation dependencies exist between those modules. Class diagrams, layered diagrams, and work breakdown structures are examples of this class of view; and
- Execution-oriented views, which describe how the system appears at run time, typically providing one or more snapshots of a system in action. These views are useful for documenting and analyzing execution properties such as performance, reliability, and security.

A second insight is that architectural description of execution-oriented views, as embodied in most of the ADLs mentioned earlier, requires the ability to model the following as first class design entities:

- Components represent the computational elements and data stores of a system. Intuitively, they correspond to the boxes in box-and-line descriptions of software architectures. Examples of components include clients, servers, filters, blackboards, and databases. Components may have multiple interfaces, each interface defining a point of interaction between a component and its environment. A component may have several interfaces of the same type (e.g., a server may have several active http connections).
- Connectors represent interactions among components. They provide the “glue” for architectural designs, and correspond to the lines in box-and-line descriptions. From a run-time perspective, connectors mediate the communication and coordination activities among components. Examples include simple forms of interaction, such as pipes, procedure call, and event broadcast. Connectors may also represent complex interactions, such as a client-server protocol or a SQL link between a database and an application. Connectors have interfaces that define the roles played by the participants in the interaction.
- Systems represent graphs of components and connectors. In general, systems may be hierarchical: components and connectors may represent subsystems that have their own internal architectures. We will refer to these as representations. When a system or part of a system has a representation, it is also necessary to explain the mapping between the internal and external interfaces.
- Properties represent additional information (beyond structure) about the parts of an architectural description. Although the properties that can be expressed by different ADLs vary considerably, typically they are used to represent anticipated or required extra-functional aspects of an architectural design. For example, some ADLs allow one to calculate system throughput and latency based on performance estimates of the constituent components and connectors. In general, it is desirable to be able to associate properties with any architectural element in a description (components, connectors, systems, and their interfaces). For example, a property of an interface might describe an interaction protocol.