CSE681 – Software Modeling and Analysis **Spring 2013**

Architecture Concept Survival Guide

These notes are a guide you use when creating an Operational Concept Document for this class, e.g., Projects #1, #3, and #5. You will find it helpful to also look at the OCD document samples in the projects directory.

Adequate Detail:

1. Need to consider: Use Cases, Critical Issues, Partitioning, Tasks, Interactions, Events, and Views. See below for discussion of some of these.
2. Use cases need to consider all types of users and catalogue services that the system should provide for users to meet their goals.
3. If you are describing a client/server system you need to list and discuss all the expected commands and messages shared by client and server. This leads to a good description of system activities.
4. Almost always need some analysis of critical issues. Must be in the context of this system. Can't lift an analysis from class notes, out of context, and plop down in report, without fitting it to the context of this architecture[1].
5. Every diagram needs a caption and at least a paragraph of discussion, often significantly more (see below).

Uses:

1. Goal is to build a usage model and (occasionally) a loading model.
2. What is the context for the system?
3. Who are its principle users?
4. What do they need from the system?
5. Is there potential for follow-on applications or products?
6. Impact of uses on application design
7. Are there usage performance issues? – address the ones that make sense of the project:
    a. How many users in each category?
    b. How many requests?
    c. How big and how frequent are the replies?
        1. How many files?
        2. How many bytes?
    d. How much data stored and retrieved?
    e. How much data searched

Issue Analyses:

1. Identify an issue.
2. Demonstrate that it is (or is not) a problem for this system.
3. If it is a problem, discuss its solution in the context of this architecture.
4. Repeat until there are no significant issues left to discuss.

[1] You are welcome to use any of the resources provided in class and in the class folder. But make it fit the context of ***THIS*** problem.

Critical Issues – Examples from Software Repository:

1. Performance: how many users can the system support with the proposed configuration?
    a. For Software Repository, probably not a problem unless the component server supports a large organization.
    b. When are the especially busy times:
        i. 9:00 AM to 10:30, 3:30 – 5:00 PM.
        ii. During major builds:
            1. whole system
            2. major subsystems
        iii. Just before major reviews.
        iv. Just before qualification.
        v. Final delivery.
    c. Is more than one project using the component server?
2. Security:
    a. What are the threats?
        i. Insider steals files and sells to competitor.
        ii. Disgruntled employee deletes critical files.
        iii. Competitor cracks our network and copies files.
        iv. Competitor sabotages critical files just before a delivery.
        v. Competitor monitors internet traffic.
    b. What are the defenses?
        i. Encryption (performance hit).
        ii. Limiting access on need basis.
        iii. Installing decryption on each client machine that will only run on that machine.
        iv. Password/certificate-based/biometric authentication.
3. Versioning:
    how do we manage growth of versions?
    a. Do we need to with 500 GB disks?
    b. Allow manifest to be open for additions without re-versioning. Will start to re-version only when closed. Once closed stays closed.
    c. Provide for archiving versions after a certain depth.
4. Usability (here's some examples):
    a. Versioning causes renaming. Need tool to fix-up project files so builds will succeed.
    b. Tree view in client interface: how do we populate top level without downloading a lot of files? Let server tell us children of any specified node.
    c. Finding components: need keyword search. Client should display descriptive text (embedded in manifest) for any selected component.
    d. Need index manifest to show all modules, programs, or systems, always shown in treeView (will need to partition these for very large component collections). Otherwise, there will be some components we can't navigate reach with tree view.

Discussion for each diagram:

1. Why is this diagram here?
2. What is its message?
3. What conclusions should we draw?

Discussion for Context and Activity Diagrams:

1. Relate activities to use cases.
2. Identify any difficult or complex operations.
3. Provide a fairly simple over-all activity diagram in the introduction, used to describe basic system operations.
4. Provide more detailed activity diagrams for the major processing threads.

Discussion for Package (or Module) and Class diagrams:

1. What are the responsibilities of each package (class)?
2. What are the important relationships between packages (classes)?
3. Are there any implementation difficulties for this package (class)?

Discussion for Interface Views:

1. Why is this screenshot here?
2. What does it tell us about how the system operates?
3. What does it tell us about user/system interactions?
4. Is this information or just data?

Organization:

Progress from most abstract, e.g., context diagram, uses, critical issues, package diagram, interface, to more detailed, e.g., activities, classes, events, detailed analyses.

Summaries:

Analysis is not worth much without conclusions. Write your summaries carefully. Most readers will not read the entire report.

1. Need executive summary at beginning to summarize your over-all conclusions – a summary of summaries.
2. Each section needs a summary describing the conclusions for this section. May be only a sentence or two. May be several paragraphs.

Critical Issues:

You walk into your Doctor's office, and describe your illness. Then you ask him about critical issues:

He examines you for 30 seconds, then he says:

> "You may have cancer. If you do, you will probably need radiation and chemotherapy. Your hair will fall out and you will probably die within three years."

He hands you his bill.

Do you think he did a good job?