CS 446 / ECE 452 / CS 646 – Spring 2013
Course Project Part4 – Architecture Recovery

1. Overview

For the term project, you are asked to architect, design, and implement a complex software-intensive system. The project will be completed in four stages, where in the first stage you will model the software architecture of the system, in the second stage you will provide detailed design models, in the third stage you will implement the system and present your project to the instructor or the TAs, and in the fourth stage you will reverse engineer as-implemented architecture of your system. A major part of your project grade will come from your implementation – **so you should start coding early**.

2. Architecture Recovery

For the last part of the project, you are to reverse engineer concrete (as-implemented) architecture from your source code, compare that architecture to your conceptual architecture from Part1 of the project, and explain the rationale for the **architectural drift (changes that are not in conflict with the conceptual architecture)** and **erosion (changes that are in conflict with the conceptual architecture)** that you discover in the process.

In the process of extracting the concrete architecture, you should also note the changes (if any) that have occurred to detailed design (e.g., to your class diagrams), and explain the rationale for the differences.

To extract the concrete architecture, you should do the following:

- Use the Integrated Development Environment (IDE) in which you have written the code for the project, and use its built-in features to trace the dependencies between code elements. For instance, in Eclipse you can use the "References" features – available from a context menu after right-clicking on a file or a package in the Package Explorer view – to see what code elements are referencing the current module.
    - You may also use a specialized research-oriented tool for software architecture recovery, such as ACDC. If you have plans on doing a comprehensive software

architecture recovery study (e.g., for students enrolled in CS 646) using one of the automated recovery tools, please speak to the course instructor first.

- Next, you should go through each subsystem and component of the conceptual architecture, check the dependencies of modules external to the selected component on the modules of the component itself, and note the discovered dependencies. That is, for class A, which is located in component C1, that references class B, which is located in component C2 (e.g., a method of class A calls a method of class B, or class A uses a data structure defined in class B), you should create a directed dependency from C1 to C2 at the architectural level. Furthermore, if C1 belongs to a higher-level subsystem S1 and C2 belongs to a higher-level subsystem S2, you should also create a directed dependency from S1 to S2.
- For each discovered architectural dependency, you should store the information at the code level (i.e., what specific dependencies between code elements were used to infer the dependency at the architectural level). If there are many code elements for one discovered dependency, you may include the complete listing in the appendices of the report – these appendices will not affect the page count of the report.
- Also, for each discovered architectural dependency, you should explain the rationale for its existence. For instance, a direct call from C1 to C2 may have been more efficient that an indirect call, which was originally planned in your conceptual design.
- For code elements that belong to new components or subsystems that were not identified in the conceptual architecture, you should explain the purpose of the new architectural components, and how these relate to the requirements for your software system.
- Finally, for any detailed design discrepancy, you should explain the rationale for the existence of this dependency, and show the modified class diagram (or diagram fragment for large diagrams) that indicates the changes in the detailed design.

3. **Submission Requirements**

- Deliverables: a written Software Architecture Recovery document submitted in class that includes:

   0. **Abstract**
   1. **Introduction**

      Scope and Purpose of the report
   2. **Software Architecture Recovery Process**

      Description of the process and tools used to recover the software architecture from the implementation artifacts, such as source code and script files

      Also include the listing of all the types of code relationships, such as called-by, used to infer higher-level dependencies
   3. **Description of the Architectural Drift and Erosion**

      Specification of any detected deviations from the conceptual architecture

      If you have decided to further specify your conceptual architecture based on the comments received for Part1 of your report, please note such differences

      For each identified deviation, carefully specify the reasoning behind it (e.g., more efficient coding or invocation structure)

      If there are important deviations from the detailed software design, please note those too as differences between the corresponding class diagrams (or diagram fragments if applicable)

- **Please limit your report to 10 pages maximum (strict limit).**
- **Deliverables: A printed report submitted in class that complies with the project specifications above.**

- **The deadline for this deliverable is Mon (Jul 29th) by 8:55am.**
- **No late submissions will be accepted.**