ICS 52: Introduction to Software Engineering
Winter 2004

Instructor: Dr. Richard Taylor
TA: Scott Hendrickson

Assignment 2: Architecture and Module Design

****

Issued: Tuesday, February, 3rd 2004
Due: Thursday, February 26th, 2004 (beginning of class)

SUMMARY

The Requirements Document and Acceptance Test Plan that you submitted for the last assignment was received and reviewed by your Customer. The Customer decided, for unknown reasons, to ignore your document, change focus somewhat, and work with another Requirements Engineer. Somewhat reluctantly you have since successfully negotiated a contract to proceed on the basis of the new Requirements Specification, which is included in this document. Note that there are likely some significant differences between the negotiated specifications and the original one you submitted, in form as well as in content.

Do not base your design on the requirements specification that you created for assignment 1; only designs based on this new official requirements specification will be graded.

In this Design Assignment, you are going to construct a software architecture and define module interfaces for **UniCal**.

STRUCTURE

The combination of the architecture and the module design is the deliverable for assignment 2. The printed copy of your document that you turn in for credit should include:

- A title page using a 20 point font with the following text centered vertically and horizontally

UniCal Architecture and Module Design

First_name Last_name
{Last four digits of your student ID}

ICS 52
Instructor: Dr. R. N. Taylor
Winter 2004

- Page numbers at the bottom of each page
- Font used for the document should be similar to Times New Roman (or any of the Times family).
- Major section headings in 18 point, other subsection headings in 16 point, and body text in 12 point.
- Stapled once in the upper left hand corner, no binders, no plastic covers.

The deliverable should adhere to the following structure:

- **Table of contents**

- **Introduction**, which provides a brief overview of the system from the point of view of the system architect.

- **Architecture**, which (a) explains the main architectural style used by the system, (b) why this style was chosen, (c) presents the high-level architecture, and (d) discusses the rationale for the architecture.

- **Component design**, which precisely describes each component in your design. As needed, each component should be further broken down into sub-components (modules). Each module, at whatever level, must be described in this section. Four aspects of each module must be described:

    - **The purpose of the module:** State the generic purpose of the module. Specify which part of the requirements document it addresses, if this is appropriate. (Appropriate items to address here include such topics as: What secret does this module hide? Does the module represent an abstract data type?)

    - **The provided interface:** Define the provided interface by listing all its methods with descriptive names. The definition includes the names of the methods, their parameters, and their return types. Provide a short commentary describing each of the methods (i.e. what it does).

    - **The required interface:** Define the required interface by listing all its modules/methods. Note that every one of the modules/methods listed should be described elsewhere in this document, either as part of the design which you are creating or else as an externally provided service.

    - **Constraints on the module:** E.g., performance constraints, platform requirements, and so on. Only include this part if necessary.

  Each module should be described on a separate page.

- **COMPRISES diagram**, which presents, in one diagram, the hierarchical breakdown of the **UniCal** architecture into its modules and sub-modules.

- **USES diagram**, which in one diagram relates all *elementary* modules according to the USES relation. Each module is labeled with a number indicating its level in the USES hierarchy. Required interfaces of each of the modules are connected to the provided interfaces of the other modules. Note that every part of an interface that is *required* should be *provided* by one of the interfaces to which that required interface is connected.

- **Integration Test Plan** which describes the test plan for the system after all the modules have been integrated.

GRADING

The architecture and module interface design should be clear, easy to understand, easy to implement, easy to change. A design "that just works" but is not consistent with the design principles presented in class will not get full credit. Apply the software engineering principles introduced in the lectures and found in the textbook.

The grading of this assignment will be broken down as follows:

5% for architectural style used and reason
30% for the quality of the architecture and its rationale (i.e. consistency with design principles discussed in class)
10% for COMPRISES Diagram and USES Diagram
40% for Module Specifications
15% for Integration Test Plan

The assignment counts 18% toward your final grade for the course.

Note:

- **Do not work in teams to complete this assignment**
- **No late assignments will be accepted**