CS 446 / ECE 452 / CS 646 – Spring 2013
Course Project Part2 – Detailed Design

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. Detailed Software Design

For the second part of the project, you are to create and specify detailed software design of your software system. The design needs to support the same requirements as your architecture, but it is to be specified at the level of classes, attributes, and operations.

If you have already coded parts of the system, you can reverse engineer your UML class diagrams from code (e.g., using built-in IDE support, or using separate tools).

- First, identify the typical scenarios for the important uses case of your system. For each scenario, **construct UML interaction (sequence or collaboration) diagrams**, depicting control and data flow between modules.

- Use the constructed UML interaction diagrams as the basis to **draw detailed UML class diagrams for each of the subsystems that you will implement in your architecture**. That is, you do not need to provide detailed class diagrams for third-party subsystems that you have not written or significantly modified.

- Provide one UML class diagram for each subsystem, or if feasible, combine all class diagrams into one. You can use stereotypes for the subsystem-to-class mappings, such as stereotype <<SubsystemA>> used to decorate ClassA to indicate that ClassA belongs to SubsystemA. **Use Facade design pattern to structure and organize**

subsystem interactions. For each class, specify all of the relevant attributes, operations, and constraints. Specify constraints as UML notes and write them as structured natural language, or as Object Constraint Language (OCL). For OCL constraints, see the reference notes posted on LEARN.

- Apply Gang of Four (GoF) design patterns to further refine your design. Within each subsystem, identify important structural, behavioural, and creational goals, and select and apply at least one design pattern per subsystem (if possible) to meet the identified goals. You need to **apply at least three different GoF patterns in your design**, other than Facade, and update your UML class diagrams accordingly. For the explanation of specific design patterns, see the corresponding lecture notes. For the code examples of all design patterns, see the GoF textbook or the following URLs.

  GoF Textbook: Gamma, Helm, Johnson, and Vlissides: "Design Patterns: Elements of Reusable Object-Oriented Software." 1st Edition, Addison-Wesley, ISBN: 0201633612
  Sample URL1: http://www.vincehuston.org/dp/
  Sample URL2: http://www.dofactory.com/Patterns/Patterns.aspx

- For the modules that perform key processing activities for your system, identify used data structures and algorithms, and specify them as pseudocode. If you have already written code, you can insert code instead, provided that it is adequately documented.

3. Submission Requirements

- Deliverables: a written Software Design Description (SDD) document (based on IEEE P1016-2005) submitted via Waterloo LEARN that includes:
  0. **Abstract**
  1. **Introduction**
     Scope and Purpose of the report
  2. **Software Architecture Summary**
     Summary of the software architecture defined in Part1 of the project
  3. **Identifying Modules and Control Flow**
     a. Explain the rationale used in module identification and encapsulation, and selection of applicable design patterns
     b. Describe overall control flow design, i.e., centralized vs. decentralized

c. Describe important modules for control flow, such as modules that manage or instantiate other modules, and modules that manage base processes
d. Specify external interfaces to your system, including signatures of your facade-based interfaces and specification of data transmitted across the system boundaries, e.g., XML files, networking messages, or UI commands

4. **Structural Design Specification**
   a. UML class diagrams, either per subsystem or combined for all subsystems
   b. Indicate where you have used GoF design patterns in your class diagrams
   c. For the selected classes and their methods, specify relevant data structures and algorithms as pseudocode
5. **Behavioural Design Specification**
   a. UML interaction diagrams for the important use case scenarios
   b. Specify boundary control use cases, such as system initialization, termination, and failure
   c. Address access control and provide a security access matrix indicating which actors have access to which modules within the system

- **Please limit your report to 20 pages maximum (strict limit).**
- **Deliverables: A printed report submitted in class that complies with the project specifications above.**

- **The deadline for this deliverable is Fri (Jun 21st) by 8:55am.**
- **No late submissions will be accepted.**