

CMPSCI 520
Fall 2008
Project 2.2
Assigned: November 24, 2008
Due: December 12, 2007

Goal – For your chosen PayPal App, complete a design of a system using any UML-based OOAD method.

1. Follow these steps (You may use any IDE and UML tool):
 - a. Beginning with the Use-Case Diagrams and specifications in your SRS, develop a domain class diagram using some method (RUP Use-Case Analysis, Noun Phrase Approach, Common Class Patterns, Class-Responsibility-Collaboration, etc.). The class diagram should include initial classes, attributes, associations, and operations. You may need to develop other representations (e.g., sequence diagrams or collaboration diagrams).
 - b. Merge the functional flow in the use cases and scenarios with the classes in the domain class diagram by constructing analysis-level interaction diagrams (i.e., sequence diagrams or collaboration diagrams) for each scenario in the iteration.
 - c. Identify/define/modify/merge analysis (boundary, control and entity) classes, associations, attributes, and operations based on analysis of the sequence diagrams or collaboration diagrams. Create analysis class diagrams (i.e., update the domain class diagram).
 - d. Develop analysis-level UML statechart diagrams for each class with "significant" state in Rational Software Architect (or similar tool)
 - e. Enhance sequence diagrams and UML statechart diagrams with design-level content.
 - f. Define deployment diagrams
 - g. Conduct a Review of all Diagrams and Documents for the iteration and repeat steps 2-7 until you are satisfied.
2. Schedule a Design Review with Professor Adrion **on/before December 12.**
3. Provide a short report of your design process, describing the steps, iterations and, particularly, the reviews and "stopping criteria". Include all diagrams and documents as an appendix.

Project 2.3
Assigned: November 24, 2008
Due: December 19, 2008

Goal: For your chosen PayPal App, complete the implementation and testing of a system using your choice of programming language. Follow these steps:

4. Develop a Test Plan and several (at least 3 test cases) based on your SRS. Select (adding if necessary) high-level requirements elements having functional, timing, and robustness requirements. For each of these three types of requirements specify a set of test cases designed to determine whether your Tool adheres to the given requirements. A test case should be specified in template form, where the template contains at least the following fields:
 - h. Requirement element being tested
 - i. Requirement element field being tested
 - j. Testing rationale (some sort of motivation or justification for this approach to testing the software system in this way)
 - k. Testing environment required (eg. needed hardware, software, or setup procedure)
 - l. Input(s) to the software (here it is desirable to specify a set of related inputs designed to provide good coverage).
 - m. Required output(s) (this is likely some function of the input)
5. Develop the necessary test scaffolding (drivers, stubs, etc) to **unit test** your tool (If you are programming in Java, you may find [JUnit](#) useful).
6. Specify a process (use a dataflow diagram, control flow diagram, or programming language code) for **unit testing** your Tool by executing the test cases defined above. The unit testing process will clearly be some kind of iteration through the test cases, but you should take care to specify the order in which the various test cases are to be executed. In addition, you should be sure to indicate how you will detect failures, what you will do to record failures, and any strategy you may have for altering the progress of the testing process in response to failures.
7. Provide a short report of your implementation and testing experience.
8. Groups should arrange to meet with Professor Adrion and demo your tool.