Software Engineering with Objects and Components
Group Tutorial Project: Deliverable 2
October 2006

Detailed Design and Preliminary Construction

Your teams have submitted requirements documents for your facets of the Secure Submission System. The second project deliverable takes this forward to begin the construction process for your proposed system. The main elements for the submission for deliverable 2 are:

1. A prioritization of the use cases you generated in Deliverable 1 of the project selecting those you want to concentrate on in your software construction process. You should include a short justification for your decisions.
2. A class diagram augmented with appropriate attributes and methods.
3. A documented change history for your use cases and class diagram.
4. Sequence or communication diagrams that demonstrate how your class diagram (suitably augmented with appropriate attributes and methods) realises the prioritized use cases. You should document the mapping from use cases to sequence or communication diagrams.
5. Activity or statechart diagrams that describe workflows.
6. Java implementation of your selected classes (this should be agreed with your tutor).
7. Appropriate tests for non-trivial classes in your implementation. An outline of how you would test the integration of your selected classes and a plan of how you would propose to test the integration of your design with the designs of the rest of your tutorial group.

Deadline

The deadline for the deliverable is:

Deliverable 2: 3pm Monday, 4th December

Submit your work to the ITO, JCMB, Room 1502.

Documents to submit

Your team should submit: an assessment proforma that will be available on the course web page, and the documentation specified above. Each element of the deliverable is described in more detail below.

1. **Prioritized Use Case Model.** You should provide a (potentially) revised version of the use case model from your first deliverable. You should take into account any feedback (from your tutor) and modify the use cases appropriately. Note that any change should be recorded in a change history. Then, in discussion with your tutor, you should identify a small subset of the use cases that you think makes a sensible basis for constructing a first implementation release. If possible, this set should be coordinated with the other teams in your tutorial group.

2. **Augmented Class Diagram.** For those use cases you have identified as a priority you should carry out more detailed design involving the enrichment of your class diagram with attributes and methods and a collection of sequence or communication diagrams, which correspond to the use cases. Note that any change to the initial design should be recorded in a change history.

3. **Change History.** Any change to the initial use case and class diagrams should be recorded in a change table.

4. **Sequence or Communication Diagrams.** Sequence or communication diagrams should illustrate in detail how a particular use case can be realised by a system structured according to your class diagram.

5. **Activity or Statechart Diagrams.** Activity or statechart diagrams that describe workflows (i.e., dynamic aspects of your system design and implementation).

6. **Java Prototype.** Given your prioritized set of requirements, choose (non-trivial) classes in your class diagrams as candidates for implementation. *You should confirm your choice of classes with your tutor.* You should choose classes that are linked in the sense that in at least one of the use cases the classes are required to collaborate to generate the correct result. Moreover, these classes may collaborate with other classes implemented by the other teams.

   You must provide Java classes for the classes selected from your class model. These classes should be involved in realizing the use cases you have identified as a priority. Your goal is to produce a working prototype for the selected classes. The Java classes you define should pass the tests you have defined as part of this practical.

7. **Testing.** Having determined the classes you intend to construct, define unit tests for these classes and implement them in an appropriate context (e.g., you may want to use the JUnit framework). The tests should be traceable back to a particular use case and should also be traceable to the appropriate sequence or communication diagram. The tests should be reasonably comprehensive. The documentation for the tests should specify what the code is testing and what the expected outcome is for each test. The test suite should comprise a collection of Java programs (i.e., test cases) that exercise a given class.

   The justification for the adequacy of the test suite is an important part of this deliverable. You should write a short narrative justifying the design of your tests and how they relate to the appropriate collaboration diagrams.

   Integration Plan. You should include a plan on what testing is required to assure the integration of your classes into the overall design. This should specify what the integration test should be directed towards and how it should be tested.

Assessment

Marks will be awarded by team, with individual team members receiving a proportion of marks in relation to their contribution as for deliverable 1. Marks for this deliverable constitute 50% of the practical marks for this course. The marks for this part of the practical constitute 12.5% of the overall assessment of this module.

Massimo Felici
October 31, 2006