Worksheet 3

Spring Semester, 2010-2011

Notes on Creating the Analysis Phase Class Diagram

J.J. Collins and Anne Meade

Steps

1. **Data Driven Design** – assumes that use cases for a package are in the same use case diagram.
    a. Underline noun and noun phrases in each use case description in one diagram
    b. Apply heuristic to eliminate poor candidates
    c. The remaining candidates are entity classes
2. Repeat for each use case: use case realisation to create a use case class diagram (fragment of overall class diagram). See Agate Case Study A3.
    a. Sketch a collaboration diagram
        i. What entity objects are necessary to support/realise the use case?
        ii. Sketch informally and include in the submission.
    b. Use a communication diagram to validate the collaboration diagram
        i. Add the entities identified in the collaboration diagram.
        ii. Add a boundary object as a token representative of the User Interface (UI).
        iii. Add a control object. What is its purpose?
            1. Control objects are sometimes compared to generals in an army who issue instructions to the foot soldiers who do the work. Control objects help to minimises dependencies between entity objects thus maximising their reuse potential. Instead of entity A calling entity B, the control object call A, and then calls B.
            2. The operations in a control object are specifications of the work being carried out by the use case. Some refer to this specification as a low level workflow. While enduring domain concepts (entity objects) rarely change, these low-level workflows referred to previously are subject to frequent maintenance. Thus control objects localise the site of change i.e. make maintenance easier.
        iv. Model messages being passed between control, boundary and entity objects
            1. Every step in the use case description corresponds to one or more message being passed in the communication diagram.
            2. Try to minimise direct message passing between entities objects.
        v. Sketch informally and include in the submission.

   c. Check using Class Responsibility Collaboration (CRC) cards. Too many responsibilities indicates possibly poor cohesion. Too many collaborators might also signal too much dependency.
   d. Draw the fragment of the class diagram resulting from (b) above. This is referred to as a use case class diagram.

3. Integrate use case class diagrams to yield a first draft of the analysis class diagram. Some authors refer to this as the conceptual class model.
   a. Add attributes to classes and specify visibility.
   b. Check that associations and multiplicities are modelled and correct.
4. Produce a second draft of the analysis class diagram – see Agate Case Study A4.
   a. Allocate classes to packages in the high-level architectural diagram.
   b. Identify opportunities for generalisations.
   c. Are abstract classes required near the root of the inheritance tree?
   d. Identify opportunities for polymorphism
   e. Specify the operations that should be listed in interfaces, possibly adding pre and post conditions using Object Constraint Language (OCL).
      i. This is normally done by the software architect when defining the high level structural architecture using a package diagram, created at the earliest possible point in the project
   f. Ensure that the diagram supports the concept of "programming to interfaces, not implementation".
      i.

In Select Factory 7.1

- Entity class is a business class
- Control class is a business class
- Boundary class is a user class
- Add in stereotypes for boundary, control, and entity classes.

Notes for Project.

1. You should realise two or three use cases.
2. You can use "common sense" to identify classes necessary to support the remaining use cases.
3. One may then assume that if time permitted, one would repeat the steps outlined above!