# 15-413: Introduction to Software Engineering

## Jonathan Aldrich

## Assignment 6: Requirements Modeling

**Due: Wednesday, October 12, 11:30am (hardcopy at beginning of class)**
30 points

This assignment is a group assignment. Each project group should turn in one response to each part, with all the names of the group members.

Parts of this assignment depend on interaction with clients. Occasionally, you may not be able to meet with clients for reasons outside your control, e.g. they are traveling. If this is the case, contact the instructor or TA to get an extension on the assignment. However, you are expected to contact your client promptly so as to avoid any possible delays.

**Requirements Modeling**

In this assignment, you will build each of several different kinds of models discussed in class: goal models, information models, sequence diagrams, and statechart models. Four-member groups must turn in one of each model; three-member groups may omit the sequence diagram.

You may use any tool you wish to develop the diagrams, or you may draw them ***neatly*** by hand. Powerpoint or Visio are natural choices for the goal model, while Omondo EclipseUML is recommended for the UML models. You can download Omondo EclipseUML from http://www.omondo.com/. A license key that can be used on any computer with a CMU IP (for academic purposes only) is available on Blackboard in the Course Information section. To install the license, copy the license file to the following directory: %ECLIPSE HOME% \plugins\com.omondo.uml.core_1.6.0

All of the models below should be related to your project in some way. If due to the nature of your project there is nothing relevant to model, contact the instructor for an alternative way to do the assignment.

Accompany each model with a short discussion covering topics such as (a) what you are modeling, (b) why modeling that might be useful for requirements, (c) how you chose the level of abstraction (what to include and what to leave out), (d) anything that is not obvious about the model elements. Turn in a copy of the model itself, together with your discussion.

## 1. Goal Model

Build a goal model that decomposes one of the highest-level goals of your project. Use parallelograms to represent goals. Distinguish between goals that your system is to achieve and assumptions that some other entity (e.g. another system, a human, the way the world works) is responsible for achieving. Break the highest-level goal into parts and decompose these recursively until they are detailed enough that (1) you could formally define the goal if you wanted to (we won't require this) and (2) the goal can be assigned to a particular entity in the software system or (for assumptions) the environment. Show as domains pieces of information that exist in the environment and in the machine, and distinguish graphically between these two kinds of domains. Show how each goal and assumption relates two or more "observed" and "controlled" domains, using dashed arrows for the controlled domains and dashed lines for the observed domains.

You may use your discretion in choosing what part of the system to model, and at what level of abstraction to model it. However, your model should be interesting enough to have at least 8 goals/assumptions and 4 domains.

## 2. Information Model

Build an information model showing either the state of concepts that your program uses, or else the data it uses as input or output. You should use UML class diagram notation to express your model. However, as this model is focused on requirements, you should ***not*** describe a model of the actual or planned classes in your program (except by coincidence).

Your information model should include classes with attributes (but not necessarily operations), and associations between classes annotated with appropriate multiplicities. You should use aggregation instead of association to represent part-whole relationships, and if there are concepts that have a kind-of relationship you should use inheritance.

You may use your discretion in choosing what part of the system to model, and at what level of abstraction to model it. However, your model should be interesting enough to have 8-10 classes or so, and should illustrate either aggregation or inheritance in addition to associations.

### 3. Sequence Diagram

Draw a sequence diagram illustrating an extended scenario of use of your system. This may incorporate a single story, multiple stories, or part of a story. As this model is focused on requirements, you should ***not*** describe a model of the actual or planned sequence of method calls in your program.

Use actors to represent various users of the system, and use components to describe the parts of the system itself and other computer entities with which it interacts.

You may use your discretion in choosing what part of the system to model, and at what level of abstraction to model it. However, your model should be interesting enough to have at least 3 actors/components and 8-10 arrows.

### 4. Statechart Model

Build a statechart model showing either how your program interacts with the outside world, or else models some element of the outside world that your program must deal with. As this model is focused on requirements, you should ***not*** choose a purely internal component of your program to model; that would be a design model.

Your statechart model should illustrate a use of multiple constructs including nested states, initial states, and at least one of history states or AND-states (simultaneous / parallel state machines as indicated by a dashed line) and preferably both (but this may depend on the domain). You may use your discretion in choosing what part of the system to model, and at what level of abstraction to model it. However, your model should be interesting enough to have 15-20 states or so.