The UNIVERSITY of VERMONT

Benjamin Fenster
CS-100 (Damon)
6 April 2003

Notes – Unified Modeling Language (Continued)

I. Aggregation
   a. Drawn Robot ———<> Team
   b. "These are the parts of this."
   c. Eg: Robots on Team, Workers in Group, etc
   d. The key is that it's easy to talk about the Robot outside the team. It is independent, but it's also "a part of."
   e. The Robot can also be parts of other things too.

II. Composition
   a. Drawn Menu ———✍ MenuBar
   b. Here it's only meaningful to talk about a Menu as part of a MenuBar.
   c. It's also true that the Menu isn't part of anything else.

III. State Charts
   a. Finite State Machines
   b. See CS100-30-9
   c. Describes a limited number of states and how to get from one to another.
   d. It's easy to write code that implements a finite state machine.
   e. States can be nested
      i. See CS100-30-10
      ii. The top-level cares about only a few states.
      iii. Within one of those there may be a number of sub-states.
   f. Parallel States
      i. Completely independent
      ii. See CS100-30-11
   g. States can get very complicated, like banking at an ATM.
   h. Usage
      i. Often used for embedded software.
      ii. User Interfaces make good sense as state machines sometimes too.

IV. Collaboration Diagram
   a. What instances exist in a single execution?
   b. instance : className
   c. See CS100-30-15
   d. How do *instances* interact and relate, not just classes.
   e. This can also show method calls, but that's not terribly useful.

V. Sequence Diagram
   a. See CS100-30-17
   b. Shows a *possible* execution sequence.
   c. May get a different order if threads are involved.
   d. Who calls what? More importantly, *when* does it get called?

VI. Use Cases
   a. Describe how the program will be used.
   b. One case deals with just one usage.
   c. Thus, many cases are needed to describe and study a program completely.
   d. Can depict these as graphics or text.
   e. See CS100-30-20
   f. Think about the perspective of the *user's goal*, not the program's solution. (Eg: Think "Confirm the Order," not "Click the OK Button")

VII. Constraints / OCL
   a. Preconditions, Postconditions, Rep Invariants
   b. Can constrain virtually anything.
   c. The goal is to limit what's legal.
   d. Can be informal (plain English) or formal (OCL is common)
   e. Just draw a box connected to something with a dashed line for things like classes.
   f. Anywhere there's text a constraint can be added between { } braces.