Slide 1

UML — Review

- “Unified Modeling Language”.
- From originators’ Web site:

  “method for specifying, visualizing, and documenting the artifacts of an object-oriented system under development”
- From *UML Distilled*:

  “family of graphical notations, backed by single meta-model, that help in describing and designing software systems, particularly software systems built using the object–oriented style”
- Many things to many people, used in different ways.

Slide 2

UML and Tools

- As mentioned last time, there are tools for drawing various kinds of UML diagrams. Some interoperate with other tools (e.g., to generate UML class diagrams from code and vice versa).
- Most tools, alas, seem to cost money. I’ve installed some free ones that seem reasonably promising in `/users/cs2194`.

Slide 3

Types of UML Diagrams — Review

- Version 2.0 of standard includes many different types of diagrams:

  Class, Sequence, Object, Package, Deployment, Use Case, State Machine, Activity, Communication, Composite Structure, Component, Interaction Overview, Timing.
- We looked at use-case diagrams last time. A quick review of others now . . .

Slide 4

Activity Diagrams

- These model overall flow of control. Might be useful to have one of these for each use case.
- Similar to old-time flowcharts, with notation for start point, end point, selection (decision/merge), concurrent actions. Also notations for showing flow of objects through system, interruptions, which participants are doing what, etc.

Slide 5

Class Diagrams

- You're (somewhat) familiar with class diagrams these from PAD II. Can show many things / many levels of detail:
- For individual classes: attributes and operations and their visibility.
- For groups of classes: relationships, including inheritance and other kinds of associations (e.g., composition).
- Can become quite complex, but useful as a compact way to show public attributes/operations of classes, inheritance relationships.

Slide 6

Object Diagrams

- Similar to class diagrams, but represent individual objects.
- Might be useful as part of other types of diagrams?

Slide 7

Sequence Diagrams

- These provide another way of modeling how things happen. Activity diagrams focus on overall process, sequence diagrams on participants.
- Notation is in terms of participants (objects or actors) and messages between them. (E.g., object A invoking a method in object B represented as A sending a message to B and (optionally) B sending a message back.) As name suggests, notation makes overall sequence of operations clear. Also can represent asynchronous interactions.
- (I'm skeptical about these but have seen examples of their use in conference papers!)

Slide 8

Communication Diagrams

- These provide yet another way of modeling how things happen, but focusing on interactions among participants.
- Notation is in terms of participants (objects or actors) and messages between them, without sequence information.
- Simpler to draw and maintain than sequence diagrams, but less informative in some ways.

Slide 9

Timing Diagrams

- These provide yet another way of modeling how things happen, but focusing on timing and states.
- Notation is in terms of participants and states, and makes it clear how changes in different participants' states are related.
- Most likely to be useful in describing something with timing constraints (e.g., real-time or embedded system).

Slide 10

Interaction Overview Diagrams

- Similar to activity diagrams, but each "action" can be a sequence/communication/timing diagram.
- Useful as a way of representing a big picture, but examples look rather complex!

Composite Structures

- These represent relationships among classes, but in a way that captures some relationships that aren't easy to express in class diagrams.
- One type — collaboration diagrams. Often used in describing design patterns.
- (Again I'm a bit skeptical, but I've seen examples!)

Slide 11

Component Diagrams

- These are similar in some ways to composite structures diagrams, but larger-scale(?).
- Useful for modeling systems with components that might be "swappable".

Slide 12

Slide 13

Package Diagrams

- These group other diagrams — use case diagrams, class diagrams, etc.

Slide 14

State Machine Diagrams

- These provide yet another way of modeling how things happen, but focusing on states and transitions.
- Very useful for illustrating some kinds of workflow, including descriptions of protocols (e.g., TCP).

Slide 15

Deployment Diagrams

- These show relationships among (semi-?)physical components of a system — e.g., in a typical Web-based application, they would show interaction among server hardware/software, client hardware/software, firewalls, etc.

Slide 16

General Advice, Revisited

- Overall idea — standard notation for representing various aspects of software systems — seems like an obvious win. Sometimes a picture *is* worth a lot of words.
- However, one could easily get bogged down in details. So, look at examples, consider when one of these types of diagrams would add value.