Slide 1

More About UML

Slide 2

Administrivia

- I will put these slides, plus links to Web-accessible material that looks useful, on the CSCI 2194 “Useful links” page.

Slide 3

UML and Software Development

- Various models for how to build software — Fowler (*UML Distilled*) talks about "waterfall", "iterative", others.
- Four basic phases, though:
  - Requirements analysis.
  - Design.
  - Coding.
  - Testing.

Slide 4

UML Diagrams and Requirements Analysis

- Use case diagrams, and use cases in general.
- Class diagrams (conceptual perspective).
- Activity diagrams (roughly similar to flowcharts).
- State diagrams (roughly similar to state-machine diagrams).

Slide 5

UML Diagrams and Design

- Class diagrams (software perspective).
- Sequence diagrams.
- Package diagrams.
- State diagrams.
- Deployment diagrams.

Slide 6

Why UML in This Course

- For simple programs (such as homework in many courses), it works, sort of, to just focus on code — writing it, or reading it. Often there's not really a requirements analysis phase.
- For larger programs it helps to pay more attention to analysis and design. Also, large and complex systems are difficult for humans to understand. Abstracting out key features ("modeling") and representing pictorially helps. Doing this in some systematic way helps more. UML is not the only imaginable way to do this, but it's one way.
- Design projects in this course are intended as small-scale versions of "real" development project — so, a chance to practice working in groups, doing a somewhat formal analysis/design, using UML diagrams, . . .

Slide 7

Requirements Analysis with Use Cases

- Starting point for "real-world" problems — often a vague description. Need to turn this into something specific enough to be a starting point for a design. Many ways, but one that seems to work. . .
- "Use cases" — informally, stores of using a system to meet goals.
- As a running example — ATM. So, we might start one use case like this:

  Withdraw money. A customer arrives at the ATM wanting to withdraw money. The customer inserts his/her card. The system prompts for a PIN . . .

  (We might even just start out with the name of the use case, if it gives enough of an idea.)

Slide 8

Use Cases — Basic Ideas

- Actors — users of the system. Usually humans, but not always.
- Use cases — what happens when actors interact with the system. Together, the use cases should describe all the functionality to be provided.
- Each use case collects possible sequences of actions ("scenarios") relating to a goal.

  ATM example — several possible scenarios for "withdraw money", right? normal withdrawal, incorrect PIN, insufficient funds, etc.

Slide 9

Use Cases — Basic Ideas, Continued

- Probably best to start with prose — for each use case, a name, plus a short description if needed. As analysis continues, fill in details of scenario(s). One format includes:
  - Name (short but descriptive).
  - Main success scenario — sequence of numbered steps, each an element of the interaction between actor and system.
  - Extensions — alternatives for some steps. E.g., if step 2 is "customer enters valid PIN", there would be an extension 2a for "customer enters invalid PIN".

    Or can represent other scenarios as "alternatives".

Slide 10

Use Case — Example Format

Use Case — Withdrawal

Main Scenario

1. Customer places card in card slot.
2. System prompts for PIN.
3. Customer enters PIN.
4. System verifies that card is valid and prompts user . . . .

. . .

Alternative: Card Rejected

4. System determines is invalid. . . .

Alternative: Insufficient Funds

Slide 11

Use Cases — HOWTO

- Choose the system boundary (what's inside? what's outside?).

  For ATM, possibly everything that happens inside the box, or inside the box and the network it's connected to.

- Identify primary actors and goals.

  For ATM, actors are customers and service personnel; goals are . . . what?

- Define use cases.
- Draw use case diagram to summarize / present visually.

Slide 12

Use Case Diagrams — Basic Ideas

- Draw use cases as ovals, actors as stick figures, system as box enclosing use cases.
- ATM example (on board) . . .

Slide 13

Use Cases — “include” Relationships

- If multiple use cases share a sequence of identical steps — factor out common steps and use “include”.
- ATM example — **Validate Account** is common to several use cases.
- (Diagram at board.)

Slide 14

Use Cases — “extend” Relationships

- One way to provide alternatives without modifying existing use case — “extend”.
- ATM example — for **Deposit**, add **Deposit Slot Mechanism Motor Failure**.
- (Diagram at board.)

Slide 15

Use Cases — Generalization Relationships

- One way to group alternatives. Specialized versions are aware of general version, not vice versa.
- ATM example — group **Deposit Slot Mechanism Motor Failure** and **Deposit Slot Mechanism Door Failure** as **Deposit Slot Mechanism Failure**.
- (Diagram at board.)

Slide 16

Use Cases — General Advice

- Try to be somewhat uniform in how you present things, but don't get hung up on details. Goal is to communicate with other humans — fellow designers, customer.
- Can make very detailed diagrams, but probably best not to.

Slide 17

Another Example (To Try In Class) — 3194 Project from 2004

- Most of you have had the experience of "collaborative programming" using a single computer — two or more people clustered around a machine, with the ability to
  - Edit, compile, and run code, and view the results.
  - Communicate with each other — verbally and by drawing pictures on a paper or a whiteboard.

  This has many advantages in all phases of program design and implementation, including debugging.
- Your mission for this course is to design an environment that supports this kind of collaborative programming among people who are *not* all clustered around a single machine — i.e., an environment for distributed interactive collaborative programming.