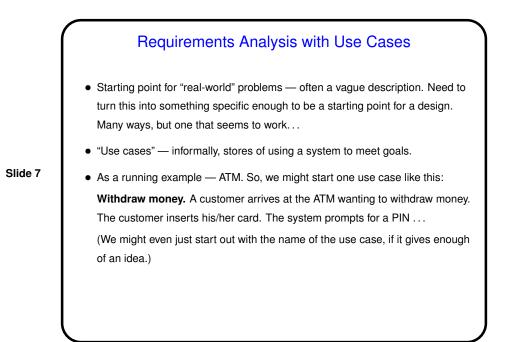


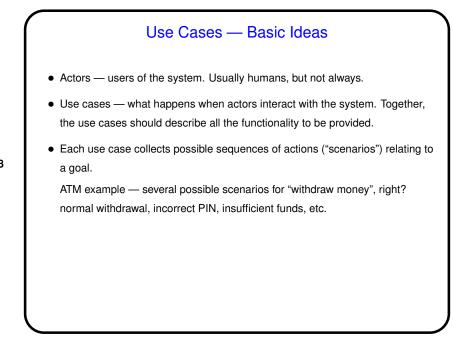
- Sequence diagrams.
- Package diagrams.
- State diagrams.
- Deployment diagrams.

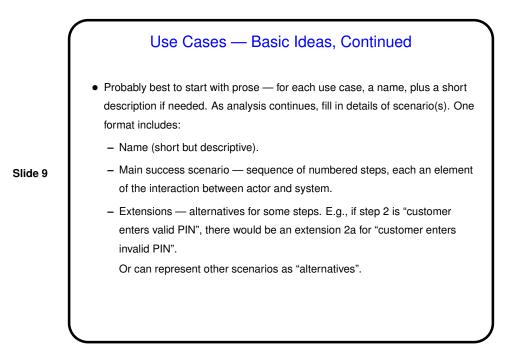


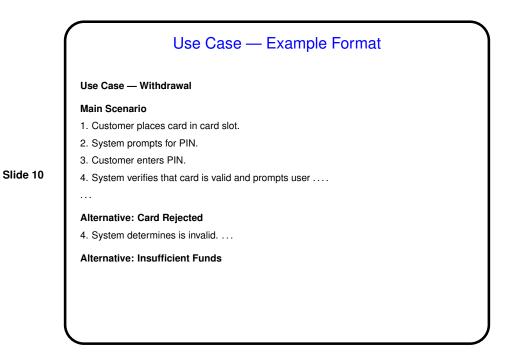
- For simple programs (such as homework in many courses), it works, sort of, to just focus on code writing it, or reading it. Often there's not really a requirements analysis phase.
- Slide 6
- For larger programs it helps to pay more attention to analysis and design. Also, large and complex systems 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, ...

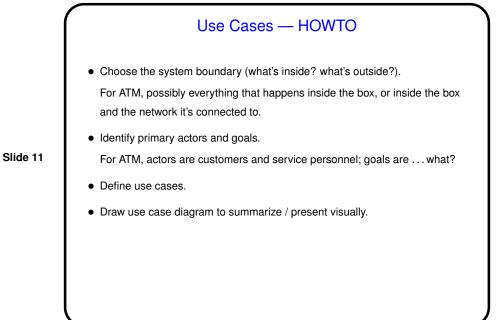


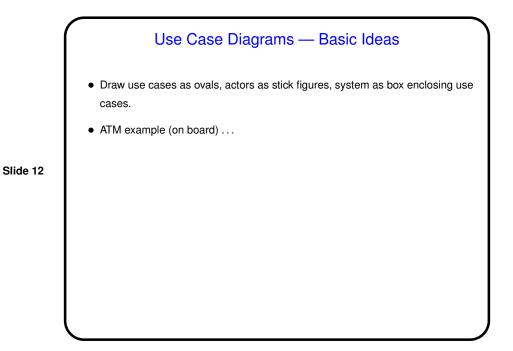


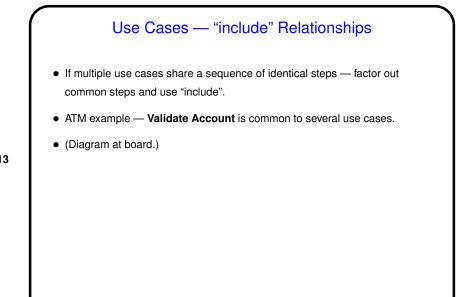


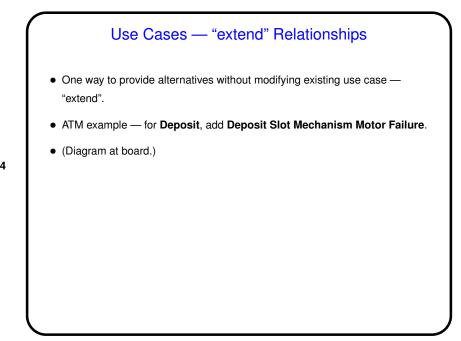


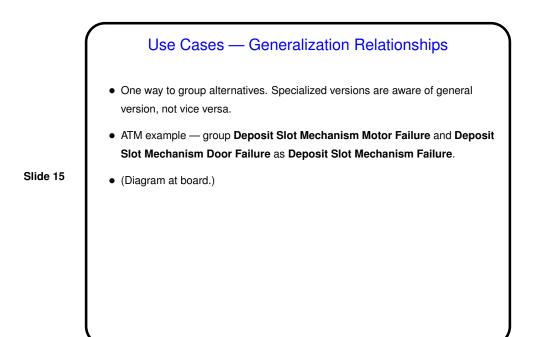


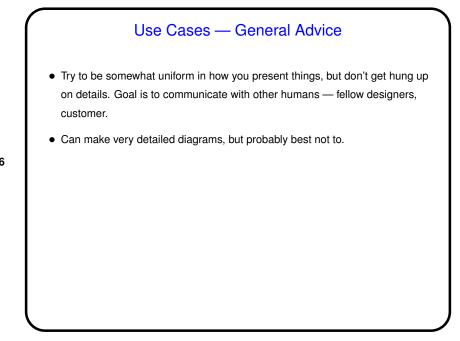












## Another Example (To Try In Class) — 3194 Project from 2004

- 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.
- Slide 17 Commur
  - 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.