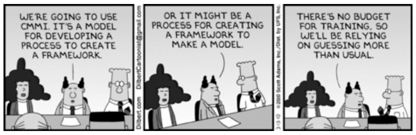
CIS 422/522

Software Life cycles and Process models
Team Assignments & First Meeting

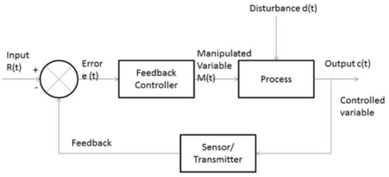

WE'RE GOING TO USE CMMI. IT'S A MODEL FOR DEVELOPING A PROCESS TO CREATE A FRAMEWORK.
OR IT MIGHT BE A PROCESS FOR CREATING A FRAMEWORK TO MAKE A MODEL.
THERE'S NO BUDGET FOR TRAINING, SO WE'LL BE RELYING ON GUESSING MORE THAN USUAL.
Dilbert.com DilbertCartoonist@gmail.com
3-13-10 © 2010 Scott Adams, Inc./Dist. by UFS, Inc.

View of SE in this Course

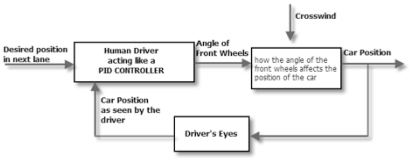
- The purpose of software engineering is to *gain and maintain* intellectual and managerial control over the *products* and *processes* of software development.
- Intellectual control implies
  - We understand the developmental goals
  - Can distinguish good choices from bad
  - We can effectively build to meet our goals
    - Behavioral requirements (functionality)
    - Software Qualities (reliability, security, maintainability, etc.)
- Managerial control implies
  - We make accurate recourse estimates
  - We deliver on schedule and within budget

Control Realities

- Reality Check:
  - Cannot fully predict consequences of our choices
  - Control is never absolute
- Implication: maintaining control is an active process (view as a feedback-control loop)



Active Control


Crosswind
Desired position in next lane
Human Driver acting like a PID CONTROLLER
Angle of Front Wheels
how the angle of the front wheels affects the position of the car
Car Position
Car Position as seen by the driver
Driver's Eyes

- Control in a software development means
  - Understand where we want to be (ideal)
  - Evaluate current delta
  - Make adjustments

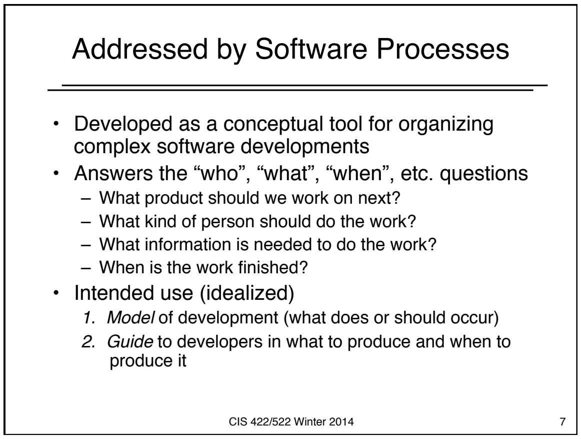
Control and Risk

- Risk: a *risk* is defined as a condition that can lead to a loss of control
  - Incorrect, misunderstood, or missing requirements
  - Poor design choices
  - Differing assumptions by developers
  - Inadequate testing, validation, etc.
- Can lead to delivering wrong product, late, over cost..
- Assessing and mitigating risk is a critical SE activity
- Assertion: well defined processes help organize work and control risks

The loss of the $500 million rocket made this possibly the most expensive computer bug in history.

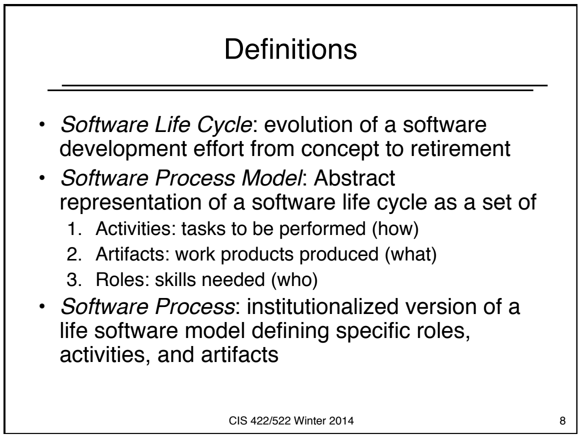
Need to Organize the Work

- Nature of a software project
  - Software development produces a set of interlocking, interdependent work products
    - E.g. Requirements -> Design -> Code -> Test
  - Implies dependencies between tasks
  - Implies dependencies between people
- Must organize the work such that:
  - Every task gets done
  - Tasks get done in the right order
  - Tasks are done by the right people
  - The product has the desired qualities
  - The product is delivered on time



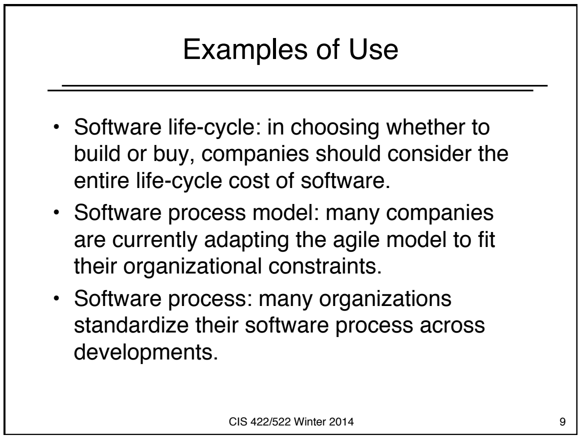
Addressed by Software Processes

- Developed as a conceptual tool for organizing complex software developments
- Answers the “who”, “what”, “when”, etc. questions
  - What product should we work on next?
  - What kind of person should do the work?
  - What information is needed to do the work?
  - When is the work finished?
- Intended use (idealized)
  1. *Model* of development (what does or should occur)
  2. *Guide* to developers in what to produce and when to produce it



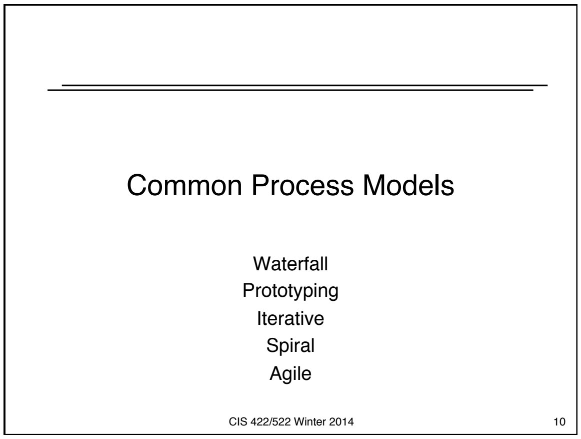
Definitions

- *Software Life Cycle*: evolution of a software development effort from concept to retirement
- *Software Process Model*: Abstract representation of a software life cycle as a set of
  1. Activities: tasks to be performed (how)
  2. Artifacts: work products produced (what)
  3. Roles: skills needed (who)
- *Software Process*: institutionalized version of a life software model defining specific roles, activities, and artifacts



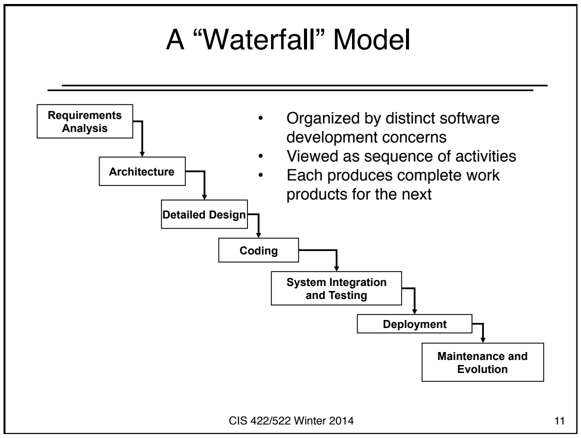
Examples of Use

- Software life-cycle: in choosing whether to build or buy, companies should consider the entire life-cycle cost of software.
- Software process model: many companies are currently adapting the agile model to fit their organizational constraints.
- Software process: many organizations standardize their software process across developments.



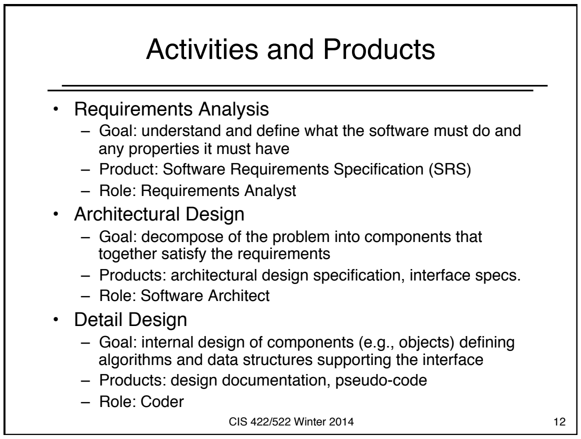
Common Process Models

Waterfall
Prototyping
Iterative
Spiral
Agile



A “Waterfall” Model

- Organized by distinct software development concerns
- Viewed as sequence of activities
- Each produces complete work products for the next



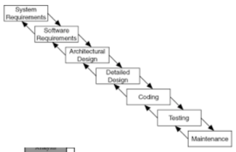
Activities and Products

- Requirements Analysis
  - Goal: understand and define what the software must do and any properties it must have
  - Product: Software Requirements Specification (SRS)
  - Role: Requirements Analyst
- Architectural Design
  - Goal: decompose of the problem into components that together satisfy the requirements
  - Products: architectural design specification, interface specs.
  - Role: Software Architect
- Detail Design
  - Goal: internal design of components (e.g., objects) defining algorithms and data structures supporting the interface
  - Products: design documentation, pseudo-code
  - Role: Coder

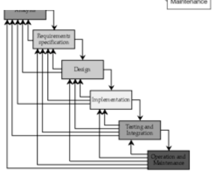
Phases and Products

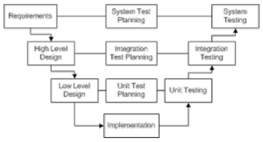
- Implementation
  - Goal: realization of the design in machine-executable form
  - Product: code
  - Role: Coder
- Integration and Testing
  - Goal: validation and verification of the implementation against requirements and design
  - Products: test plan, test cases
  - Roles: tester, user (customer)
- Maintenance (really multiple distinct activities)
  - Goal: repair errors or update deployed system
  - Products: bug fixes, patches, new versions
  - Role: Architect, Coder, Tester

Waterfall Model Variations

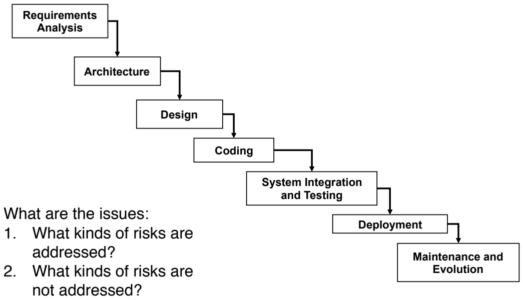


There have been many variations





A “Waterfall” Model



What are the issues:

1. What kinds of risks are addressed?
2. What kinds of risks are not addressed?

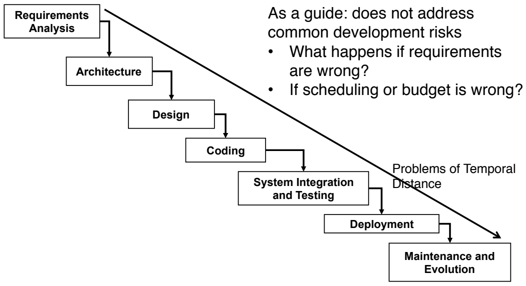
Limitations of the waterfall model

- Model implies that you should complete a given stage before moving on to the next
  - Does not account for the fact that requirements constantly change
  - It also means that customers cannot use anything until the entire system is complete
- Model implies that you can get the requirements right up front
- The entire functionality is developed and then tested all together at the end
- The model implies that once the product is finished, everything else is maintenance

A “Waterfall” Model*

As a guide: does not address common development risks

- What happens if requirements are wrong?
- If scheduling or budget is wrong?



Requirements Analysis

Architecture

Design

Coding

System Integration and Testing

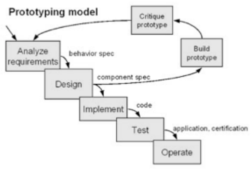
Deployment

Maintenance and Evolution

Problems of Temporal Distance

Characteristic Model: Prototyping

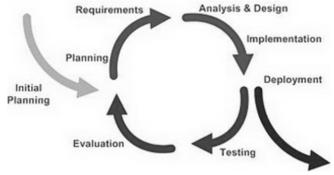
- Waterfall variation
- First system versions are prototypes, either:
  - Interface
  - Functional
- Which waterfall risks does this try to address?



Prototyping model

Characteristic Processes: The Iterative Model

- Process is viewed as a sequence of iterations
  - Essentially, a *series of waterfalls*
  - Each iteration builds on the previous one (e.g., adds requirements, design components, code features, tests)
  - Each iteration produces complete set of work products deliverable software
  - Customers provide feedback on each release
  - There is no “maintenance” phase – each version includes problem fixes as well as new features



Iterative Model

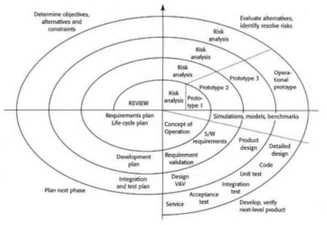
- Also called “incremental development”
- Addresses some common waterfall risks
  - Risk that software cannot be completed – build incremental subsets
  - Risk of building the wrong system – stakeholder have opportunities to see the software each increment
  - Also, can double check feasibility, schedule, budget and others issues

Advantages of Incremental Development

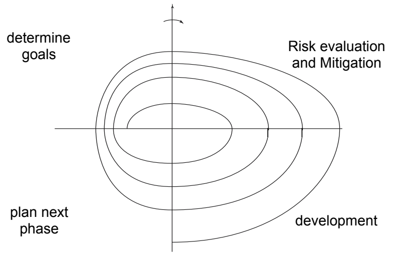
- Customers get usable functionality earlier than with waterfall
- Early feedback improves likelihood of producing a product that satisfies customers
  - Reduces market risk: if customers hate the product, find out before investing too much effort and money
- The quality of the final product is better
  - The core functionality is developed early and tested multiple times
  - Only a relatively small subset of functionality added in each release: easier to get it right and test it thoroughly
  - Detect design problems early and get a chance to redesign

Characteristic Processes: The Spiral Model

- Process viewed as repeating cycles of increasing scale
- Identify risks and determine (next set of) requirements
- Each cycle builds next version by extension, increasing scale each time

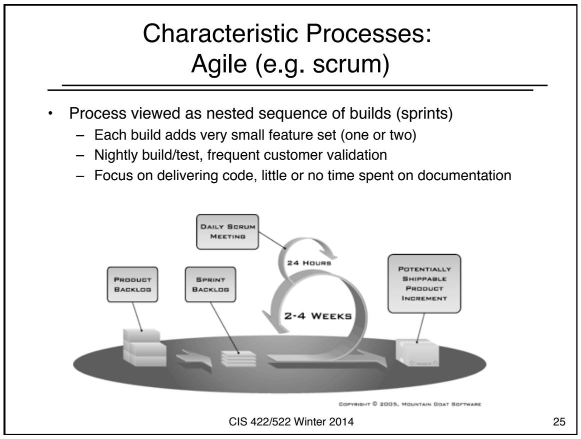


Spiral Model



Spiral Model Goals

- Response lack of explicit risk analysis and risk mitigation in “waterfall” process
- Includes risk analysis and mitigation activities at each phase (e.g., prototyping)
- Explicit Go/No-Go decision points in process

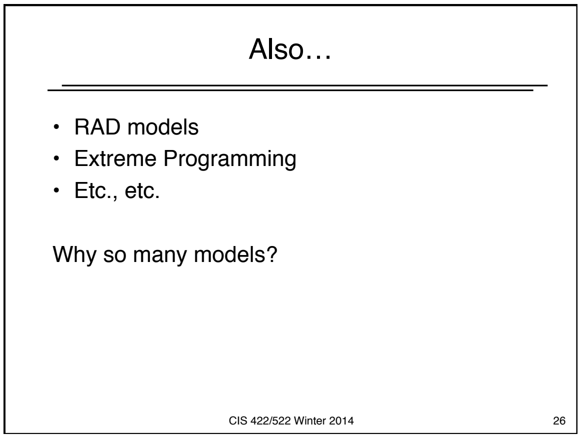


Characteristic Processes: Agile (e.g. scrum)

- Process viewed as nested sequence of builds (sprints)
  - Each build adds very small feature set (one or two)
  - Nightly build/test, frequent customer validation
  - Focus on delivering code, little or no time spent on documentation

Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment

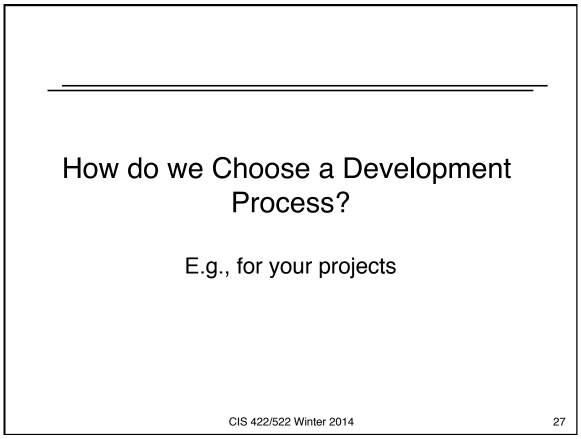
Copyright © 2005, Mountain Goat Software



Also…

- RAD models
- Extreme Programming
- Etc., etc.

Why so many models?



How do we Choose a Development Process?

E.g., for your projects

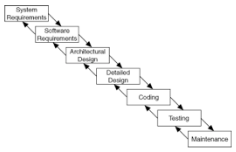
Objectives

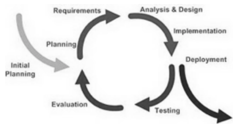
- Goal: proceed as rationally and systematically as possible (I.e., in a controlled manner) from a statement of goals to a design that demonstrably meets those goals within design and management constraints
  - Understand that any process description is an abstraction
  - Always must compensate for deviation from the ideal (e.g., by iteration)
  - Still important to have a well-defined process to follow and measure against

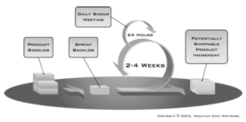
A Software Engineering Perspective

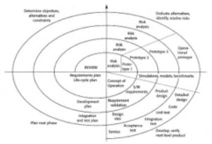
- Question of control vs. cost: processes introduce *overhead*
- Choose process to provide an appropriate level of control for the given product and context
  - Sufficient control to achieve results
  - No more than necessary to contain cost and effort
- Provides a basis for choosing or evaluating processes, methods, etc.
  - Does it achieve our objectives at reasonable cost?
  - Does it address the most important developmental risks?
- Need to agree on kind of control you need and how you will accomplish it

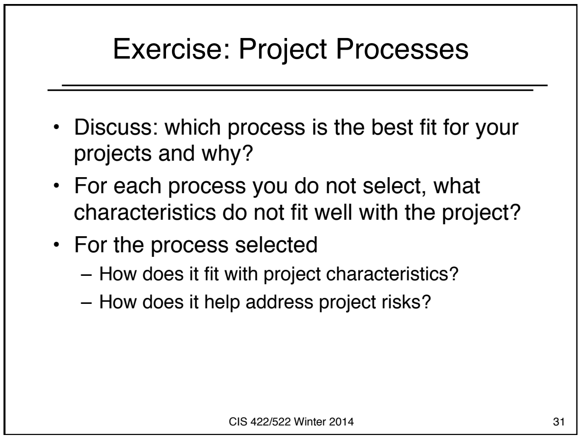
Exercise: Which Model?





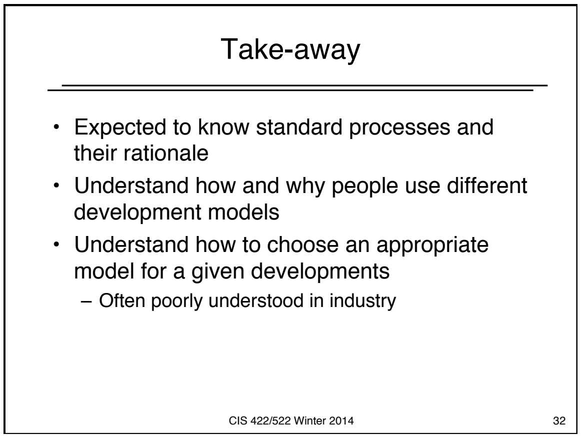

Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
Copyright © 2005, Mountain Goat Software





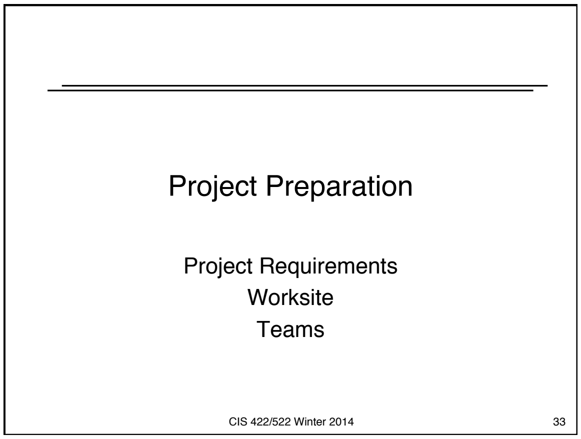
Exercise: Project Processes

- Discuss: which process is the best fit for your projects and why?
- For each process you do not select, what characteristics do not fit well with the project?
- For the process selected
  - How does it fit with project characteristics?
  - How does it help address project risks?



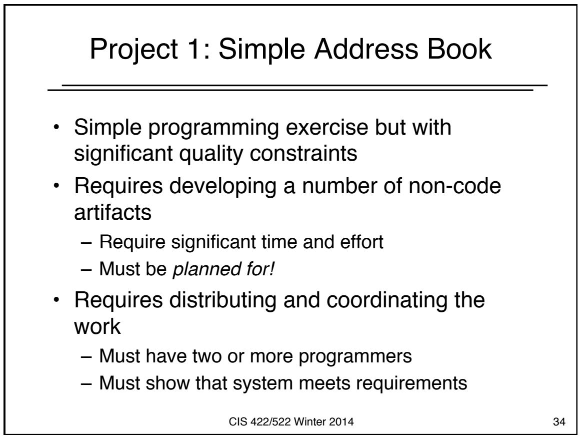
Take-away

- Expected to know standard processes and their rationale
- Understand how and why people use different development models
- Understand how to choose an appropriate model for a given developments
  - Often poorly understood in industry



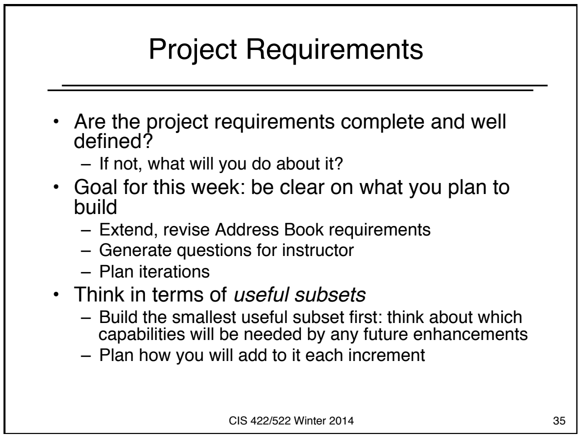
Project Preparation

Project Requirements
Worksite
Teams



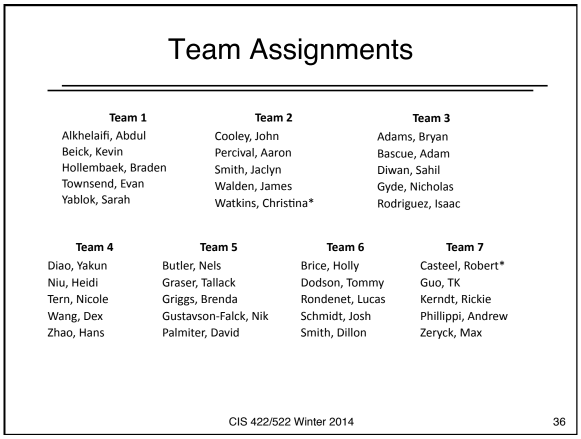
Project 1: Simple Address Book

- Simple programming exercise but with significant quality constraints
- Requires developing a number of non-code artifacts
  - Require significant time and effort
  - Must be *planned for!*
- Requires distributing and coordinating the work
  - Must have two or more programmers
  - Must show that system meets requirements



Project Requirements

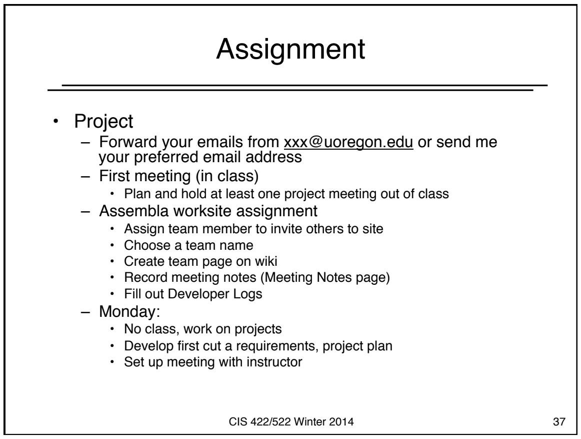
- Are the project requirements complete and well defined?
  - If not, what will you do about it?
- Goal for this week: be clear on what you plan to build
  - Extend, revise Address Book requirements
  - Generate questions for instructor
  - Plan iterations
- Think in terms of *useful subsets*
  - Build the smallest useful subset first: think about which capabilities will be needed by any future enhancements
  - Plan how you will add to it each increment



Team Assignments

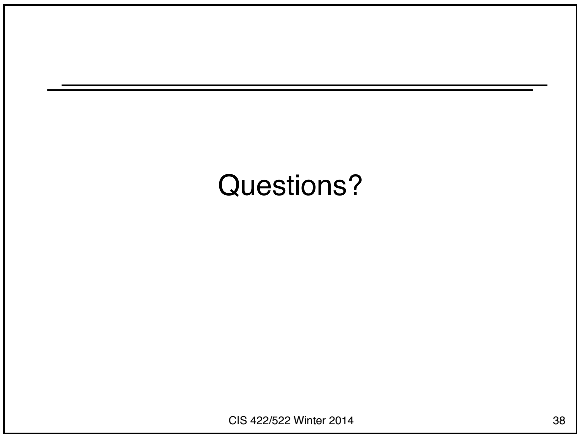
Team 1	Team 2	Team 3
Alkhelaifi, Abdul	Cooley, John	Adams, Bryan
Beick, Kevin	Percival, Aaron	Bascue, Adam
Hollembaek, Braden	Smith, Jaclyn	Diwan, Sahil
Townsend, Evan	Walden, James	Gyde, Nicholas
Yablok, Sarah	Watkins, Christina*	Rodriguez, Isaac

Team 4	Team 5	Team 6	Team 7
Diao, Yakun	Butler, Nels	Brice, Holly	Casteel, Robert*
Niu, Heidi	Graser, Tallack	Dodson, Tommy	Guo, TK
Tern, Nicole	Griggs, Brenda	Rondenet, Lucas	Kerndt, Rickie
Wang, Dex	Gustavson-Falck, Nik	Schmidt, Josh	Phillippi, Andrew
Zhao, Hans	Palmiter, David	Smith, Dillon	Zeryck, Max



Assignment

- Project
  - – Forward your emails from xxx@uoregon.edu or send me your preferred email address
  - – First meeting (in class)
    - Plan and hold at least one project meeting out of class
  - – Assembla worksite assignment
    - Assign team member to invite others to site
    - Choose a team name
    - Create team page on wiki
    - Record meeting notes (Meeting Notes page)
    - Fill out Developer Logs
  - – Monday:
    - No class, work on projects
    - Develop first cut a requirements, project plan
    - Set up meeting with instructor



Questions?