CS 370
REVIEW:
The Importance of Prerequisites
Introduction

- Prerequisites:
  - Called “upstream work”
  - Include:
    - Requirements
    - Architecture/Design
The Case for Prerequisites

- **Appeal to Logic**
  - Need to know cost
  - Know what you’re building!
  - Risk reduction

- **Appeal to Analogy**
  - Building a house
  - “Food chain” analogy → avoiding “pollutants” in system

- **Appeal to Data**
  - Early a defect is in the “food chain” → more expensive to repair
    - Error in requirements → MOST expensive to repair later
Different Approaches

- Two development approaches:
  - Sequential ("Waterfall model")
  - Iterative
- Usually project is some combination of the two

- Plan-driven $\rightarrow$ traditionally more sequential
- Agile $\rightarrow$ traditionally more iterative
Do We Need Prereqs with Iterative Projects?

- Projects without prerequisites $\rightarrow$ iterative cost < sequential cost
  - Errors detected closer to time of insertion
  - HOWEVER:
    - Discovering errors at end of iteration $\rightarrow$ still need to redesign, recode, retest $\rightarrow$ unnecessary cost
    - Costs absorbed piecemeal $\rightarrow$ average cost will be about the same

- (Iterative + prerequisites) cost < (Iterative only) cost
Which to Choose...

- **Sequential, up-front**
  - Requirements → fairly stable
  - Design → straightforward and fairly well understood
  - Dev team → familiar with applications area
  - Project → contains little risk
  - Long-term predictability → important
  - Cost of changing requirements/design/code downstream → likely to be high

- **Iterative**
  - Requirements → not well understood or unstable
  - Design → complex and/or challenging
  - Dev team → unfamiliar with applications area
  - Project → contains lot of risk
  - Long-term predictability → NOT important
  - Cost of changing requirements/design/code downstream → likely to be low