Part III of Course (Classical Software System Development)
Unified Process Vs. Waterfall

- Note that a waterfall-like process occurs in each iteration of the UP
- Basically you have to analyze before you design
- ...and design before you code
- ...even if you do this again and again over many iterations
Unified Process
Vs.
Waterfall

- So these models are not completely unrelated
- (I.e., these models are related!)
- But they are quite different in many ways
- How to know which one to pick?
- We’ve mentioned the issue, but let’s look in more detail
Waterfall Vs. Unified Process II

- Neither one is “one size fits all”
- Douglas R. Bonebrake: “I advocate methodology selection based upon project characteristics.”
- The following notes follow Bonebrake’s discussion
Different software life cycle models...
  ...are all still software life cycle models
In other words, they have more similarities than differences
  ...even though the UP sure looks different from the waterfall (when drawn in a fig.)
How are the similar?
  Let’s discuss before going to the next slide
Waterfall Vs. UP (cont.)

- You still need to do standard activities:
  - Analysis
    - (or whatever you call it and related activities)
      - requirements, specs, problem definition, domain modeling, etc.)
  - Design (what’s that?)
  - What else?
Waterfall Vs. UP (cont.)

- You still need to do standard activities:
  - Analysis
    - (or whatever you call it and related activities
      - requirements, specs, problem definition, domain modeling, etc.)
  - Design (what’s that?)
  - Development
  - Testing
  - Deployment...
“We may call these steps by various names, segment them in different phases, or use iterations to step through them...” – D. R. Bonebrake, 2010
People and organization involved must:

- Understand the life cycle model
- Be culturally compatible with the model
  - Corporate culture differs from one place to another
Waterfall Vs. Unified Process
Selecting the right one for the job

- The problem and domain is predictable and understood
  - Waterfall or UP?

- Problem and domain are poorly understood
  - Waterfall or UP?
Waterfall Vs. Unified Process
Selecting the right one for the job

- Functionalities are hard to time-order in terms of delivery
  - Waterfall or UP?

- Functionalities are easy to time-order in terms of delivery
  - Waterfall or UP?
Waterfall Vs. Unified Process
Selecting the right one for the job

- Development team is large and not collocated
  - Waterfall or UP?
- Development team is small (Bonebrake suggests <10)
  - Waterfall or UP?
- Hint: waterfall puts more emphasis on documents of various kinds
Waterfall Vs. UP – it’s not just UP

- RUP (stands for?) is a special case of UP
- UP is a special case of IID
- IID: Incremental, Iterative Development
  - RUP, UP, Agile development, Scrum, Extreme programming, perhaps Spiral models
Waterfall Vs. Unified Process
Selecting the right one for the job

- When you can’t know ahead of time exactly what will be needed
- Use UP, because waterfall requires backtracking
  - Can apply what is learned in nth iteration to next iteration
- Recall: backtracking is very expensive!
When you have to back up

Spiral Model

Five Layers or Iterations

Define
Design
Demonstrate
Develop
Deliver

Source: http://www.instructionaldesign.org/models/spiral_model.html
Source:
A Key Part of Spiral Models

- Project failure is considered possible
- If things look bad, cut it loose
- “Don’t throw good money after bad”
- Example:
  - Pascal/M programming language
- Let’s look at project failure some more
Why Software Projects Fail

Large Corporations Cringe Over Major Software Development

Why?

- Project planning and estimating is too inexact
- Project status is reported misleadingly
- System quality is too low
Large Corporation Management is Part of the Problem

Why?

- Top execs reject quality estimates
- They pressure the project schedule, reducing quality of the product
- They tend to make substantial requirements changes mid-project
  - We know that is bad, right?
Why the top management-software project disconnect?

- The problems identified did not just happen by accident
- They have “root causes”
Why is estimating/planning so “Inaccurate”?

- Estimates tend to be requested too early to get them right
- Even good understanding of the proposed project can be hard to reduce to time and money
- Demands for new requirements without opportunity to change estimates
- Not using modern estimating tools
- Careful estimates not accepted; better-sounding but overly optimistic ones used instead
Why is status reported misleadingly?

“PMs are simply not trained to carry out this important activity. Surprisingly, neither universities nor many in-house training programs deal with status reporting.” – p. 6
Why are realistic schedules not accepted, but unrealistic ones are?

- Big shots always demand more for less!
  - Large projects typically require over 3 years
  - PMs have a hard time defending estimates people at the top don’t like
  - Lack of historical data to translate project understanding into time and money
  - Schedule is overly affected by external considerations
Why do Requirements Change Mid-Project?

“The root causes of requirements changes are dynamic businesses. Real-world requirements for software must change in response to new business needs. However, average change rates of 2 percent per month indicate... analyzing... requirements... should be improved.” – p. 7
Why is Quality Control Poor?

“Effective software quality control is the most important single factor that separates successful projects from delays and disasters.” – p. 7

“... finding and fixing bugs is the most expensive cost element for large systems.” – p. 7

“More than 50 years of empirical studies have proven that projects with effective quality control cost less and have shorter schedules than similar projects with poor quality control. However, a distressing number of PMs are not aware of the economics of quality control.” – p. 7