Quiz 1 and Exercise …

What did you learn from yesterday’s meeting …

- Summarize the main things you learned
- Do you have an understanding of the problem to be solved?
- Do you have an idea of what any of the major features are?
- Do you have an idea of what technologies to choose from?
Steps

1. Start with a general idea of what to build
2. Code like hell
3. Maybe use some combination of (optional)
   - informal design, debug/test methods
4. Stop when you have a product ready to release

What are disadvantages?

- assessing progress
- assessing quality
  - if you find halfway through your design is flawed …
    usually have to start over (rework)
- identifying and mitigating risks
What are *advantages*?

- no overhead ... just pure coding!
  - classic mistake: upstream mistakes 10-100x more expensive to fix downstream
- can show signs of progress early (unless team “goes dark”)
- requires little expertise (just need to program)
- good for bug-fix type projects

How it works …

- progress sequentially through steps
- review held after each step (to advance)
- if mistakes, backtrack and redo
- “assembly line” development
Waterfall Model

The waterfall model is document driven

What are **advantages**?

- Can find (some) errors early on
- Tasks easier because of (forced) stable requirements
  - assume they don’t change, but if they do have to “backtrack”
- Helps to ensure high-quality (but large overhead)

Waterfall Model

The waterfall model is document driven

What are **disadvantages**?

- no software until late in the project
  - low “visibility” for actual software (few signs of progress)
- hard to correctly capture all requirements at start for most projects
- excessive documentation (can waste time, unless required)