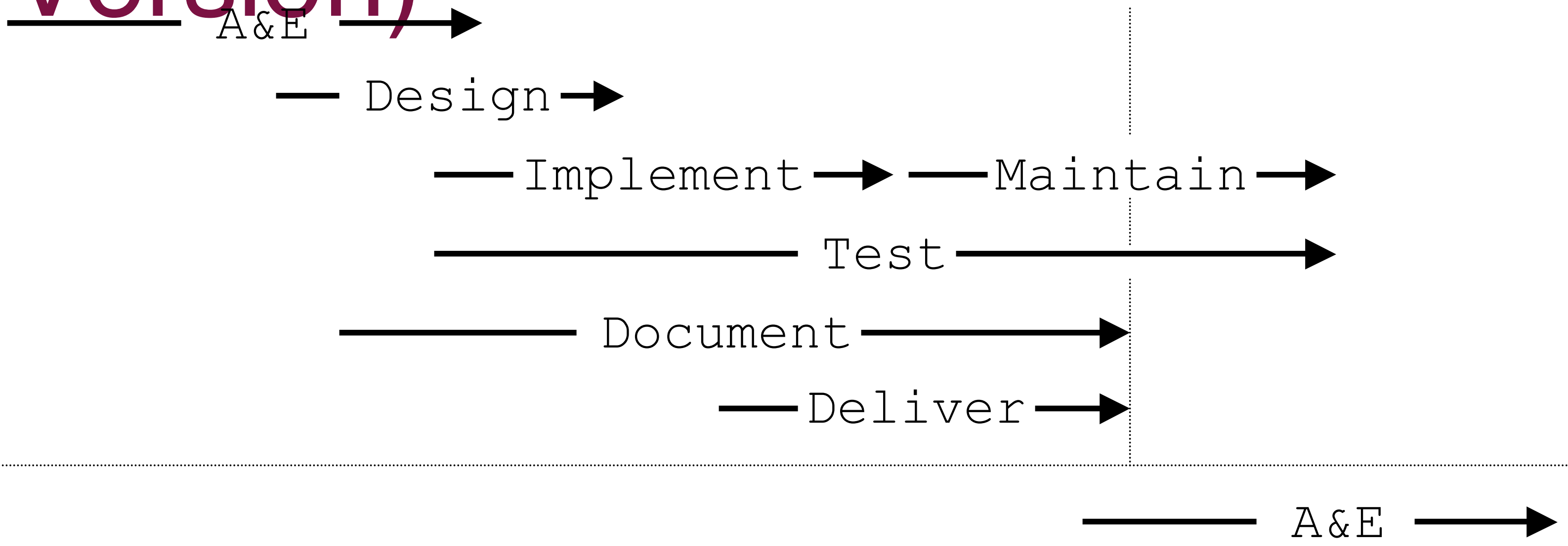
A Real-World Development Lifecycle

Greg Wilson
gvwilson@cs.utoronto.ca

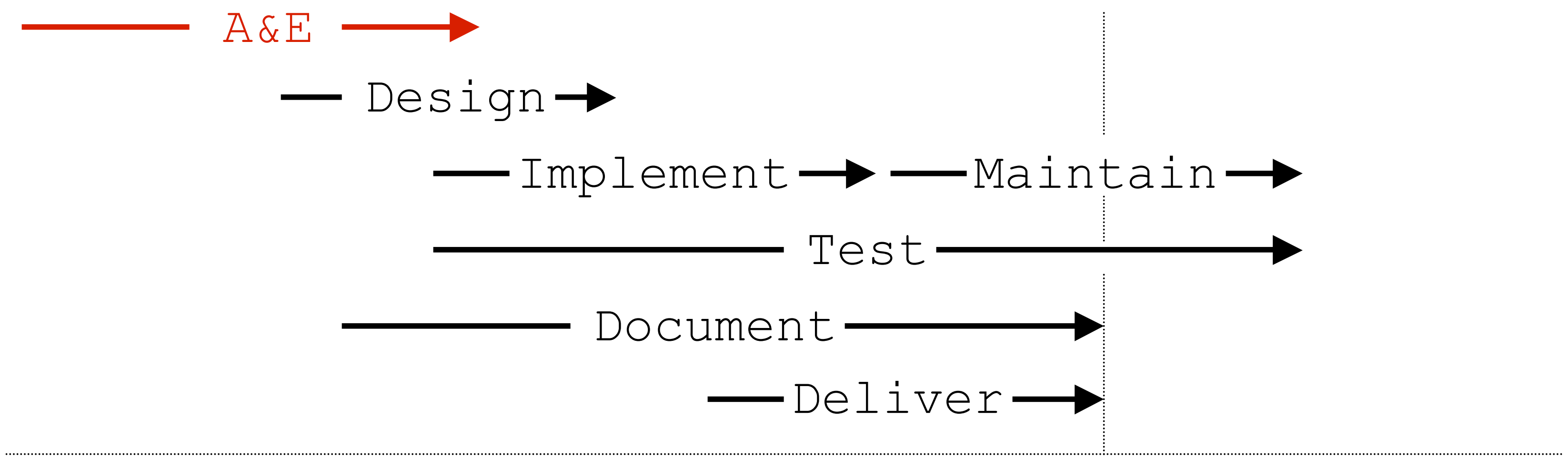
Context

- A lot of noise about methodology these days
- But:
  - Very little hard data
  - And most developers don’t work at the extreme ends of the spectrum
- Here’s a process that delivered on time, on spec, on budget for five years running
  - Worked at Nevex/Baltimore/HP
  - Working right now for one-term student projects

Project Lifecycle (Industrial Version)

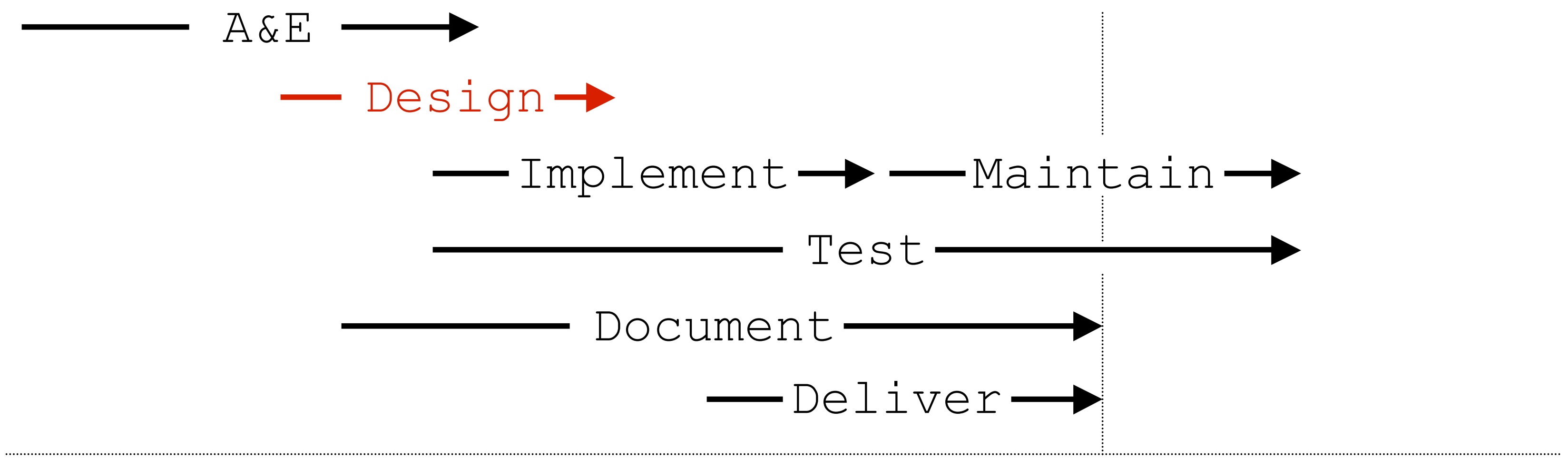


Analysis & Estimation



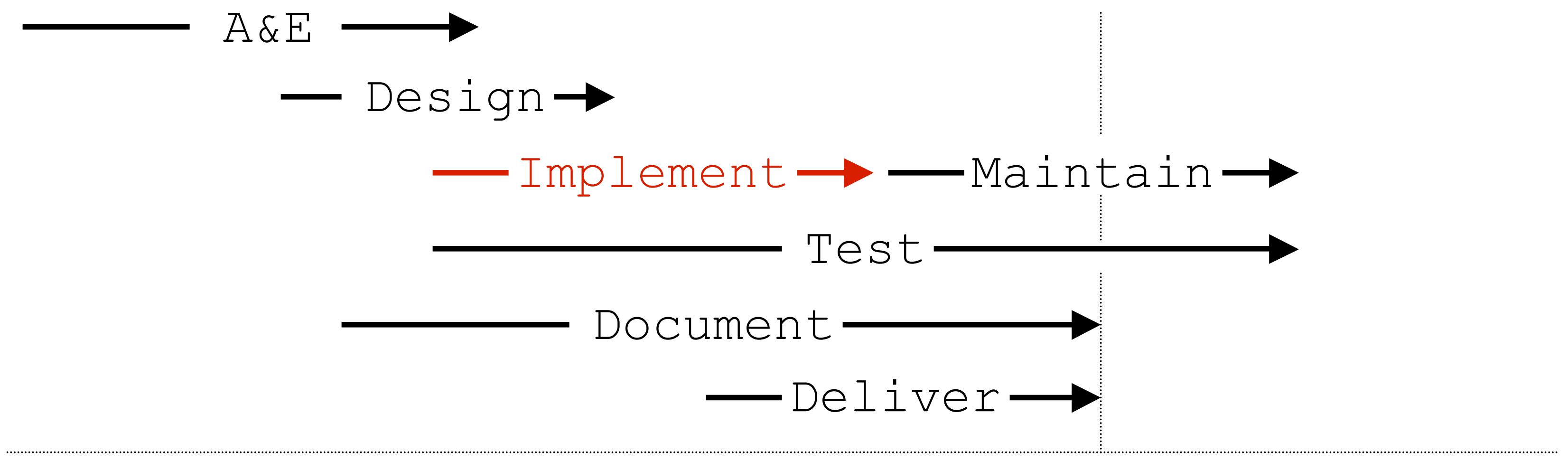
- Everybody does analysis & estimation (A&E)
  - The most important part of a successful project
  - Capabilities ⇒ Features ⇒ Schedule
  - More on this later

Design



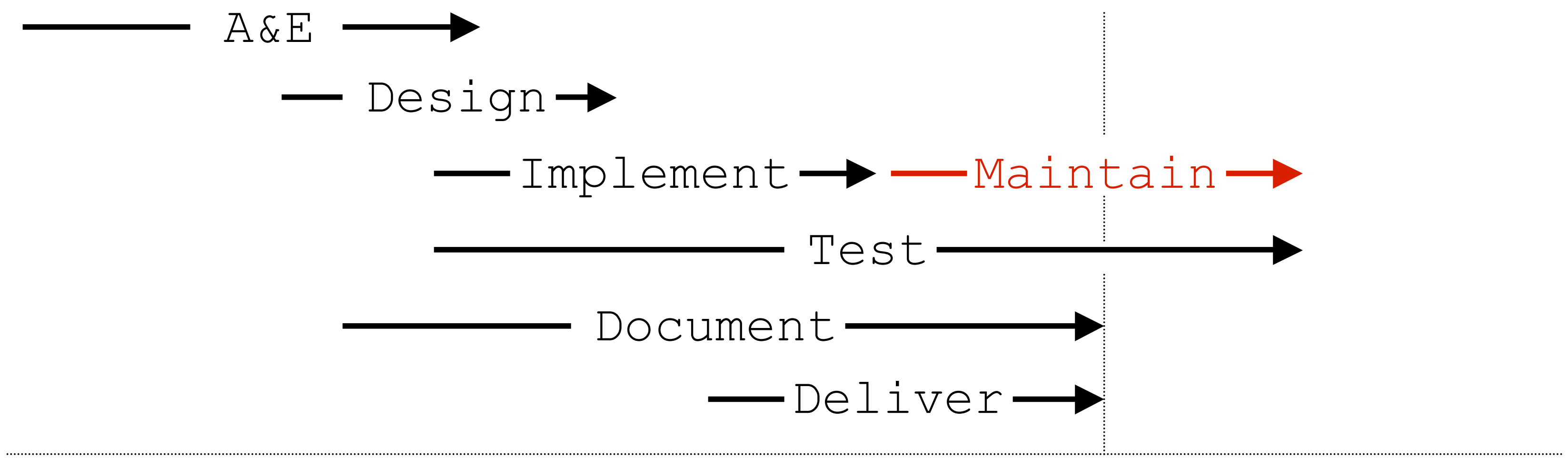
- Have to know *how* in order to figure out *what*
  - "…and then a miracle occurs"
  - Build throwaway prototypes to check
- No point creating write-only designs

Implementation



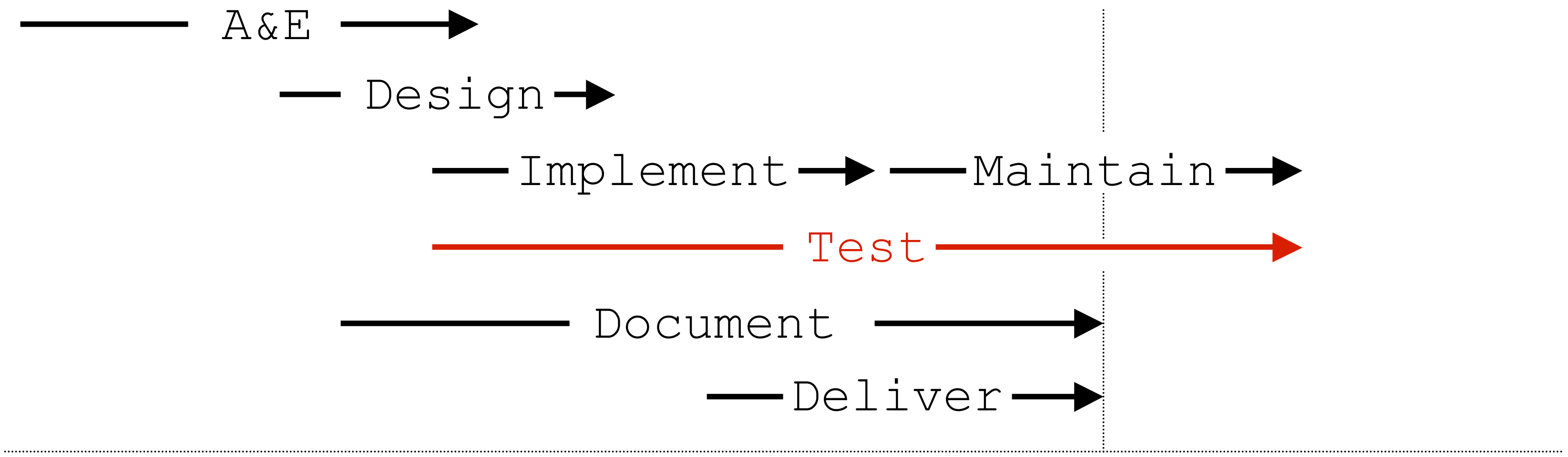
- Notice how short this segment is
  - 50% of overall time is typical
- Keep schedule up-to-date
  - The fine art of cutting corners

Maintenance



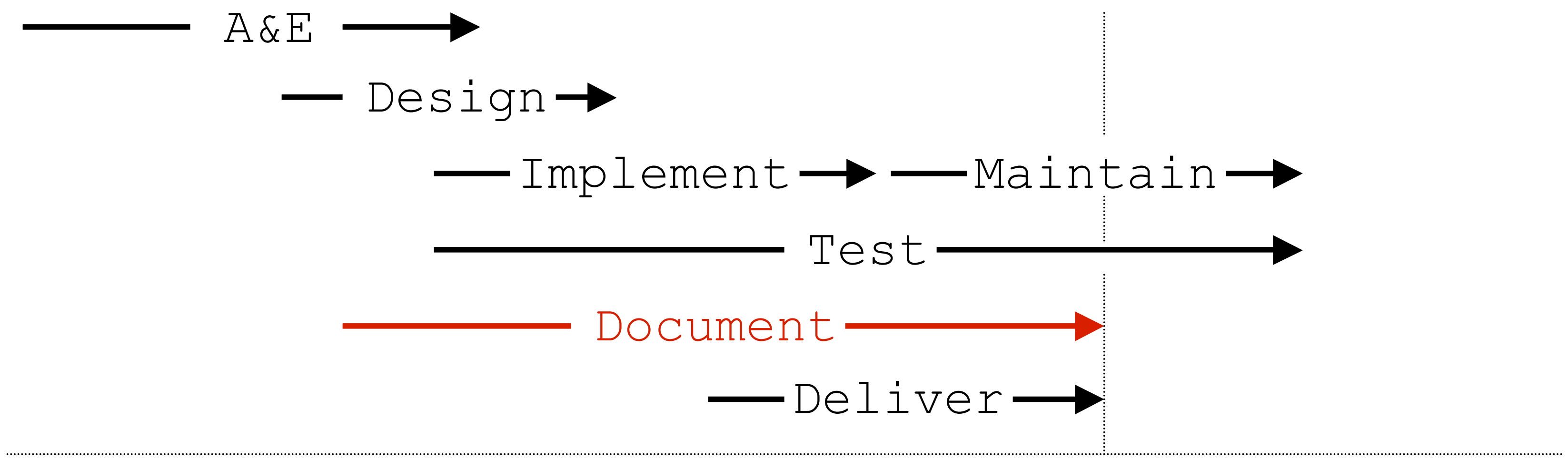
- Stop adding features weeks or months before release
  - QA & docs need a stable target
- That *doesn't* mean you stop coding/testing

Testing



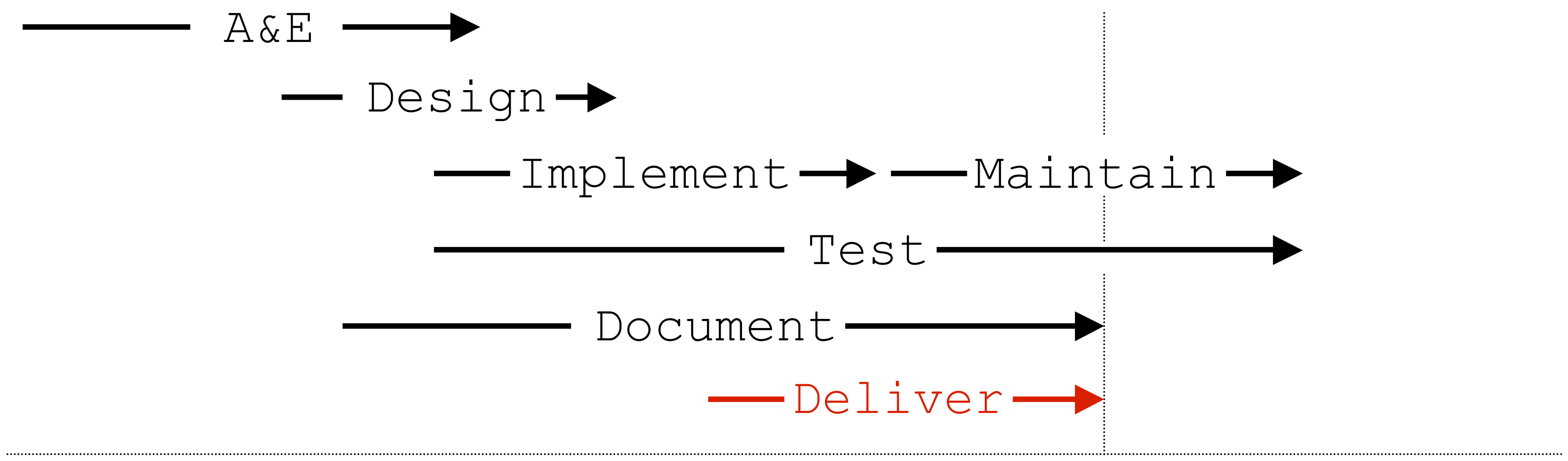
- Test-first coding only works for small things
- QA must be involved in design
  - Don’t build things that can’t be tested
- Keep testing as long as there’s a customer

Documentation



- Get technical writers involved in design
  - Often better at laying out GUIs than engineers
  - If they can't explain it, you don't build it
- Keep them involved (they hate surprises)

Delivery



- Writing an installer is a programming problem
  - Often as hard as adding new features
- Build and test installers well before shipping
  - QA should only work with released installer

The Planning Game

- Product manager lists desired capabilities
- Developers analyze and estimate features
  - Estimates include testing, documentation, deployment, etc.
  - Total typically three or four times longer than time available in schedule
- Product manager then chooses which features will actually be built
  - Rank each low/med/hi on time and importance
  - Is *not* allowed to shave developers' estimates

Afterward

- Always do a post-mortem
  - What went right, what went wrong
  - Can't improve what you don't think about
- Label and branch your code
  - Fix on one branch, add to the other
  - Often move code back and forth
- Start the next round of A&E
  - Often overlaps final stages of previous project

And Now, the Academic Version

- Student projects are typically one term long
  - 13 weeks
  - 8-10 hours/week
  - Lots of other demands on their time
- Usually have little or no experience with the subject matter
- How to tailor this lifecycle to those needs?

Thirteen Weeks

- Warmup exercise (2 weeks, 10%, individual)
  - Throwaway code that forces students to learn tools and technologies
- Analysis & estimation (2 weeks, 10%, group)
  - Result is a target and a schedule
- Development (7 weeks, 30%, group)
  - Everyone tests along the way!
  - Cut corners early!
- Final report (2 weeks, 50%, group)

Project Environment

- IDE with symbolic debugger (Eclipse)
- Version control (CVS, Subversion)
- Build & test (Ant, JUnit, CruiseControl)
- Issue tracker (CVSTrac)
- Archived mailing list (Mailman)
- Schedule (continuously updated)
- See Trac (http://projects.edgewall.com/trac)

How Well Does It Work?

- Pretty well, actually
  - Employers and other profs like the results
  - Time management
  - It’s not done ’til it’s done
- Most students schedule too optimistically
  - But learn quickly
- Now focusing on multi-term projects
  - More emphasis on architecture docs
  - More opportunity for students to lead teams

Things to Remember

1. Tools are signposts, not destinations
2. Collaboration skills are more important than coding skills
3. A week of hard work can sometimes save you an hour of thought
4. The deadline isn’t when you’re supposed to finish; the deadline is when it starts to be late
5. Code unto others as you would have others code unto you

http://pyre.third-bit.com