CSE 403

Software Lifecycle
Fall 2012

September 26, 2012

Showing Up

(Tip of the Day)

- "Eighty percent of success is showing up"
- Show up for work every day/class unless you have a good excuse (or make one up :-)) -- sick, appointment, vacation
- Let your boss know
- www.cs.washington.edu/403->I'm skipping class

Software Lifecycle

A "software lifecycle" is the process by which an organization delivers software.

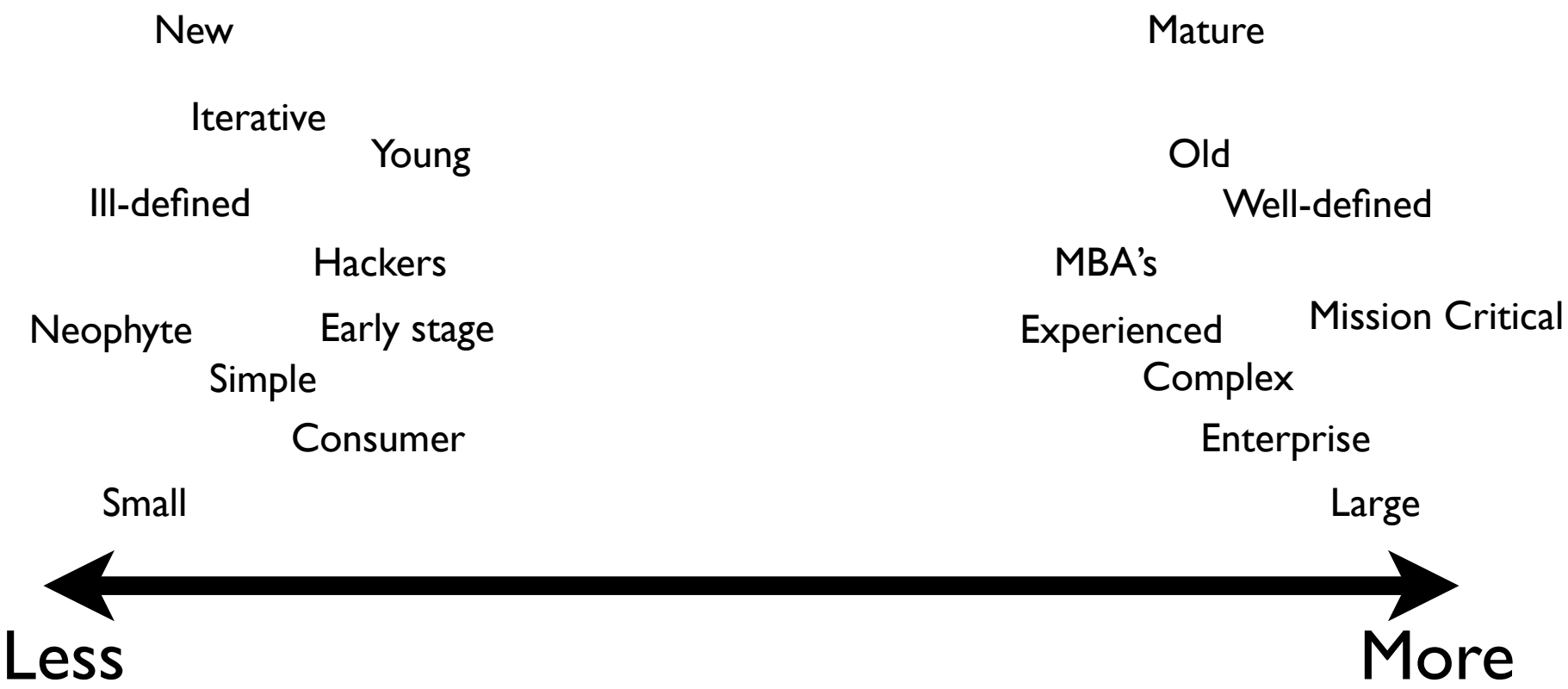
There are many kinds of lifecycles

- Software engineering is more than a "mere matter of programming"
- Type of software
- Size/complexity of software
- Maturity of company/software
- Lack of planning (process has a "life of its own")

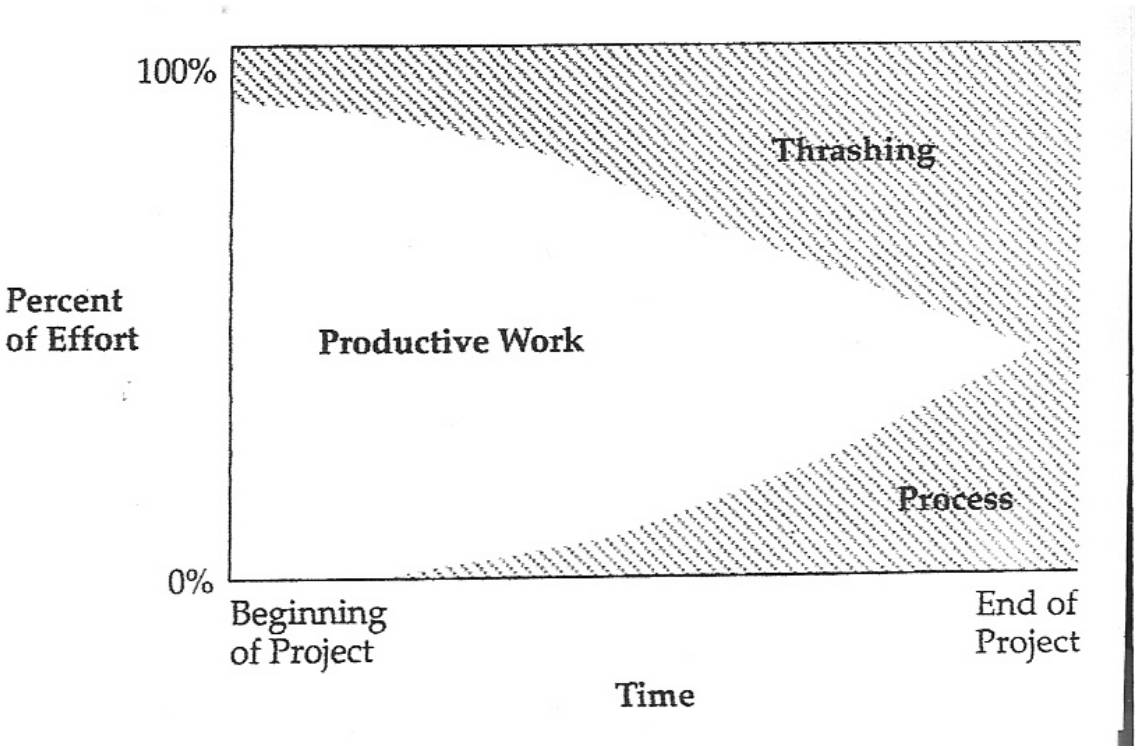
Software size/complexity

- Big/many requirements -- “Boiling the ocean?”
- Unclear requirements/unchartered territory -- “You don’t know what you are doing”
- Lots of features
- Many people
- External issues
- Lines of code, number of classes

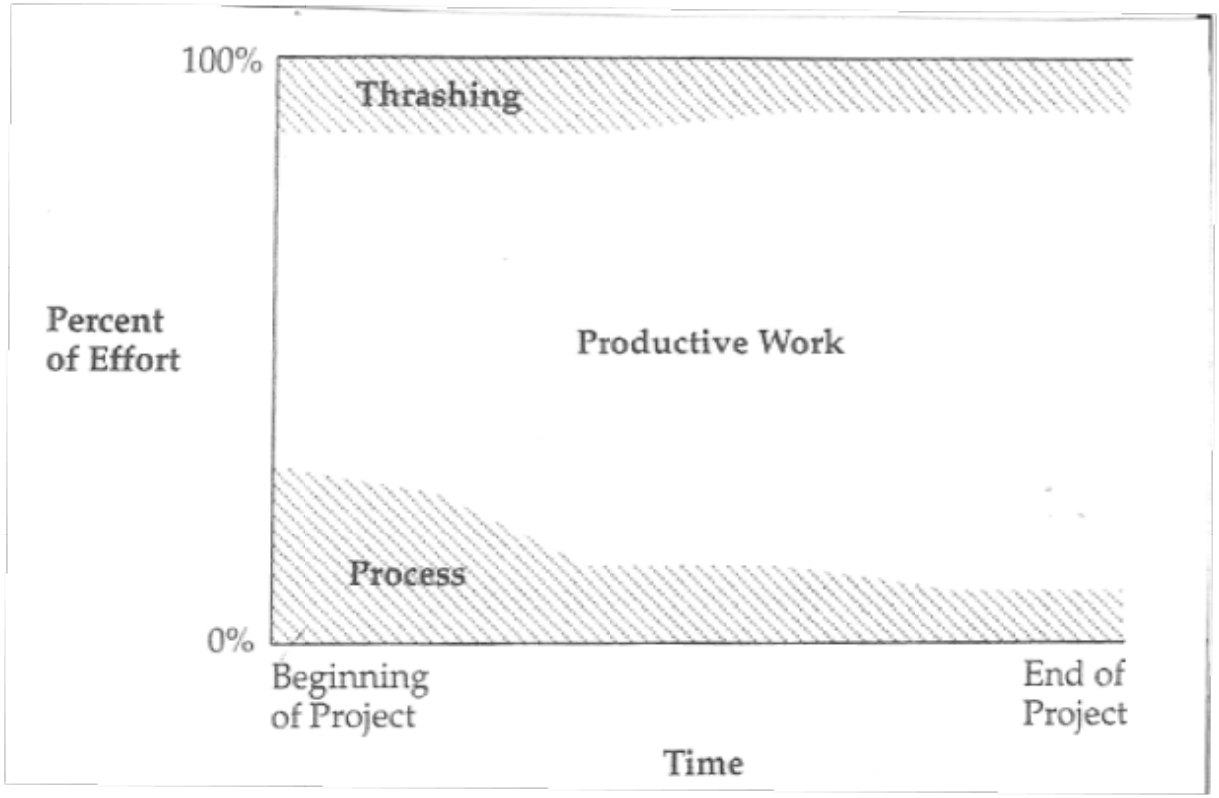
Amount of process



Not enough attention to process(?)



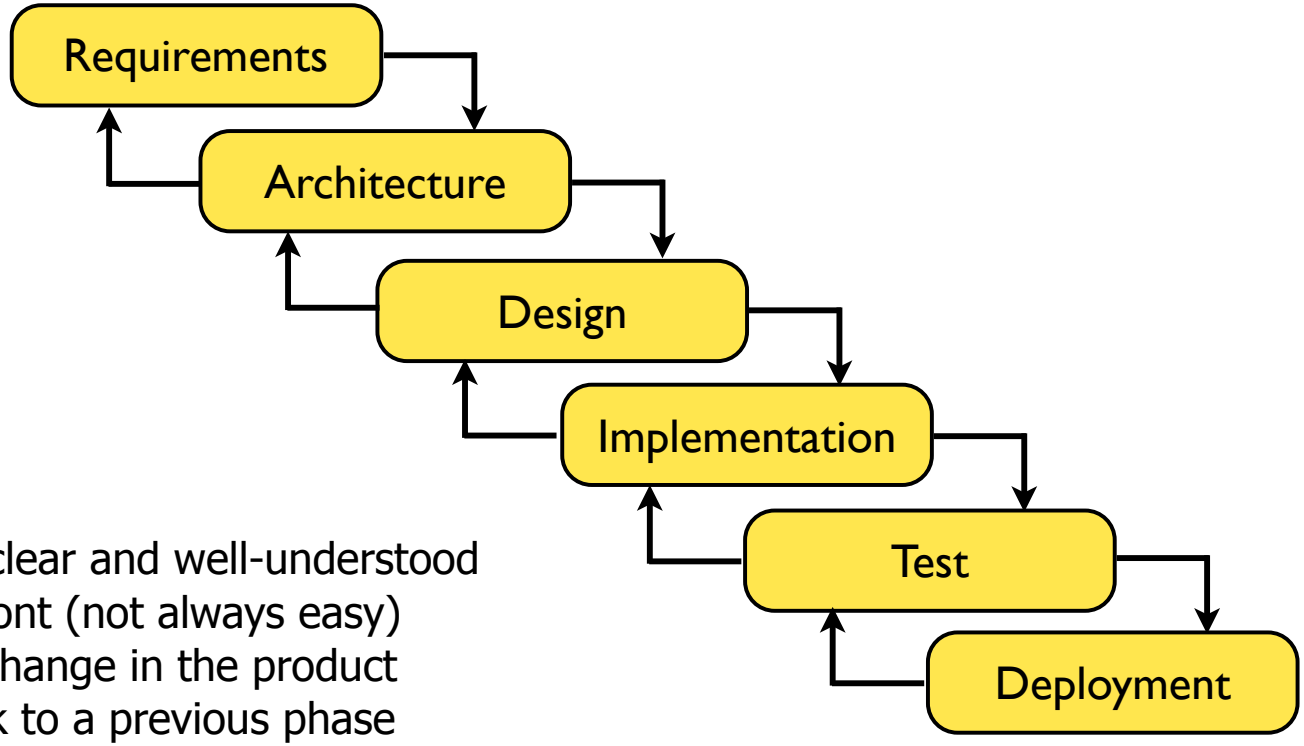
Early attention to process (and tools)



Lifecycle Tasks

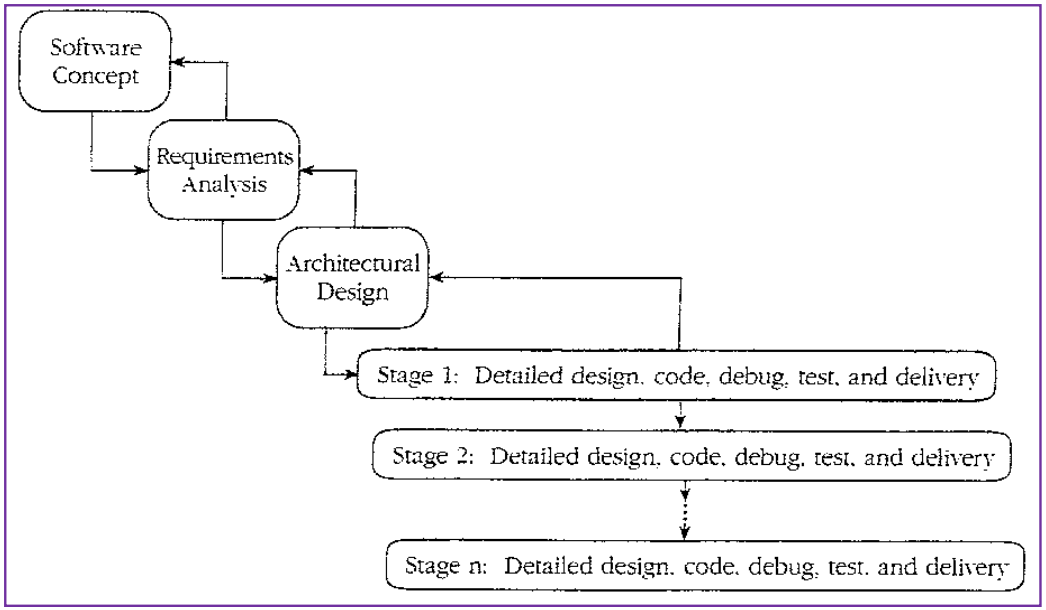
Requirements	What
Architecture	Big Picture
Design	Details (but...)
Implementation	Code
Test	Make sure it works
Deployment	What the customer sees
Support/Maintenance	Keeping it working
Compatibility	Old versions work with new

Waterfall Model



- assumes requirements will be clear and well-understood
- requires a lot of planning up front (not always easy)
- rigid, linear; not adaptable to change in the product
- costly to "swim upstream" back to a previous phase
- hard to “pipeline”
- nothing to show until almost done ("we're 90% done, I swear!")
- out of vogue, in 2012

Staged delivery model



- Waterfall-like beginnings
- Then, short release cycles: plan, design, execute, test, release, with delivery possible at the end of any cycle

Ad hoc development

Great for early stage development for a small team.

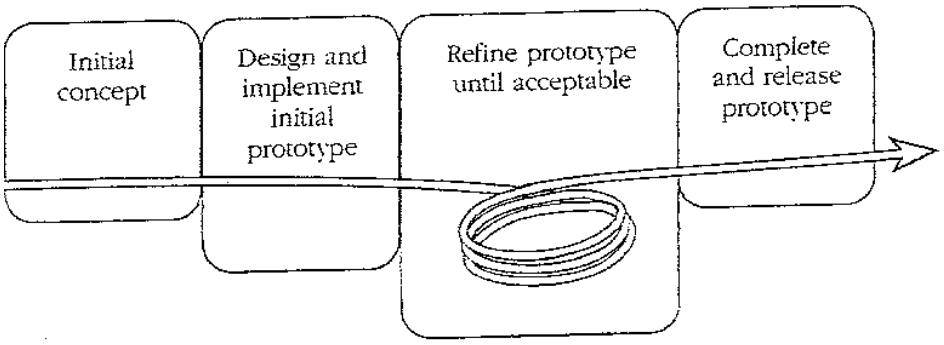
- Get early feedback quickly
- Efficiently deploy a lot
- Low overhead

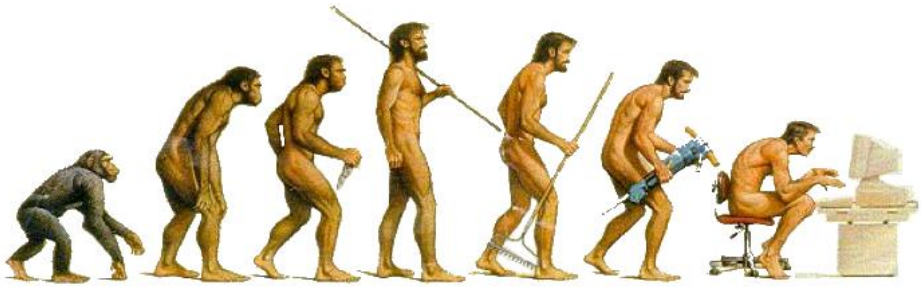
...but not without down sides

- Are you building the right thing?
- Will it scale (across multiple dimensions)?
- Susceptible to disasters
- Progress grinds to a halt...
- Not "engineering?"

Evolutionary prototyping model

- Develop a skeleton system and evolve it for delivery
- Staged delivery: requirements are known ahead of time
- Evolutionary: discovered by customer feedback on each release





Agile Development

agile software development: An adaptive, iterative process where teams self-organize and build features dynamically.

- Extreme Programming
- Scrum

values:

- **Individuals and interactions** over processes and tools
- **Working software** over documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan