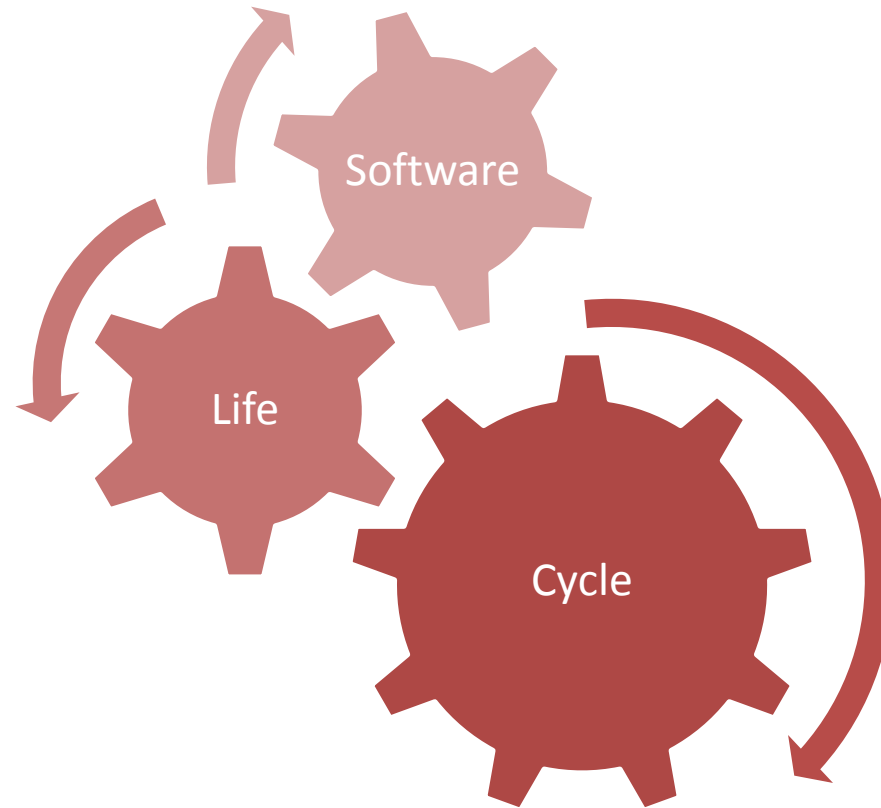


Software Development Lifecycle

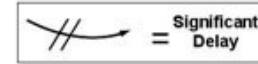


thinking about the process

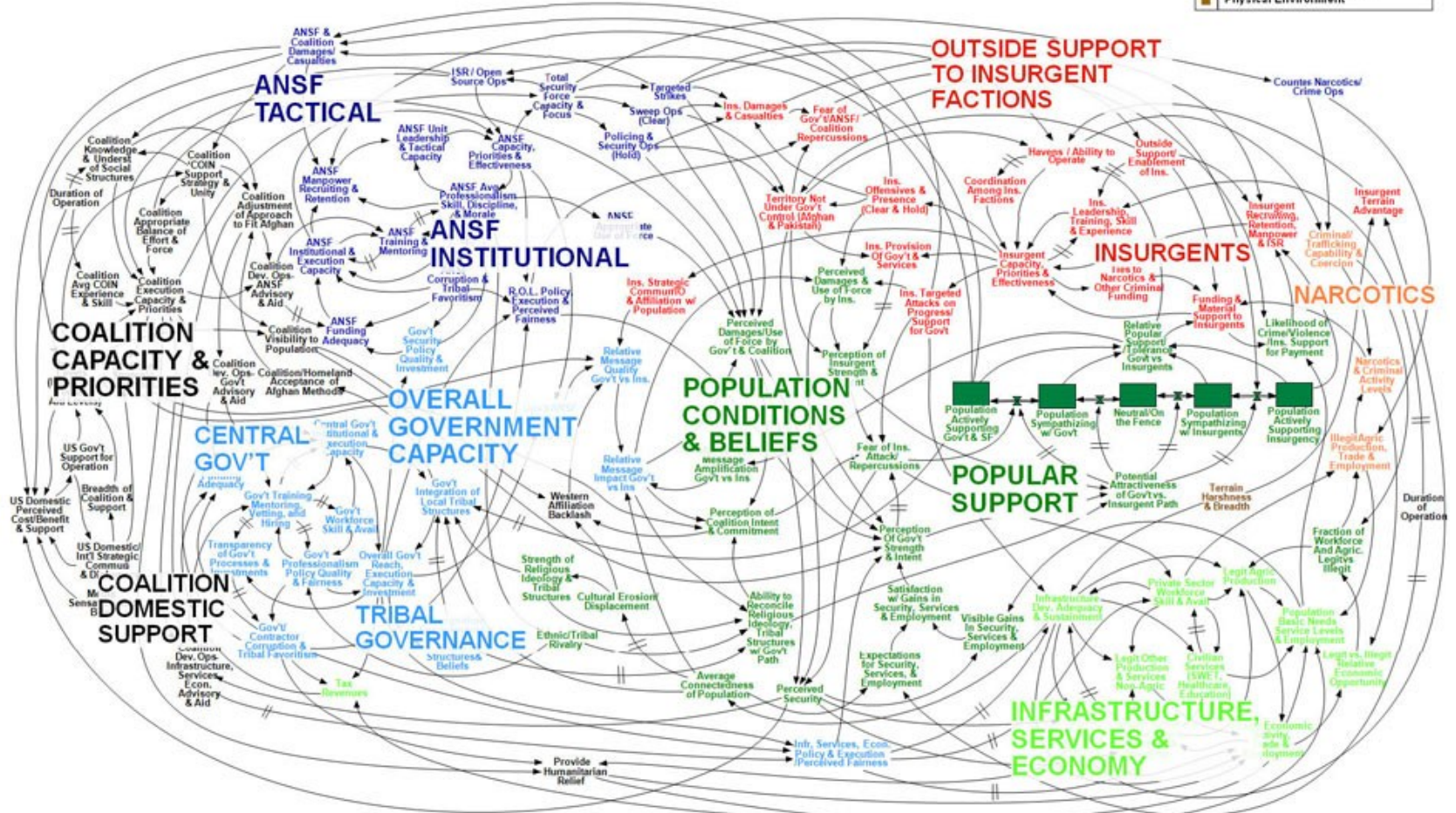
How complex is software?

What is complex?

Afghanistan Stability / COIN Dynamics



- Population/Popular Support
- Infrastructure, Economy, & Services
- Government
- Afghanistans Security Forces
- Insurgents
- Crime and Narcotics
- Coalition Forces & Actions
- Physical Environment



WORKING DRAFT - V3

How complex is software?

- Measures of complexity:
 - lines of code
 - number of classes
 - number of modules
 - module interconnections and dependencies
 - time to understand
 - # of authors
 - ... many more

How complex is software?

- Measures of complexity:

- lines of code

Windows Server 2003: 50 MSLoC

- number of classes

Debian 5.0:

324 MSLoC

- number of modules

- module interconnections and dependencies

- time to understand

- # of authors

- ... many more

How big is 324 MSLoC?

- 50 lines/page \Rightarrow 6.5M pages
- 1K pages/ream \Rightarrow 6.5K reams
- 2 inches/ream \Rightarrow 13K inches
- 13K inches \approx twice the height of the Allen Center
- 5 words/LoC @ 50 wpm \Rightarrow 32M min \approx 61 years

And we don't just want random words,
we want compiling code!

Managing software development

- Requirements
- Design
- Implementation
- Testing
- Maintenance

Outline

- Why do we need a lifecycle process?
- Lifecycle models and their tradeoffs
 - code-and-fix
 - waterfall
 - spiral
 - staged delivery
 - ... there are many others

Ad-hoc development

- Creating software without any formal guidelines or process
- Advantage: easy to learn and use!
- Disadvantages?

Ad-hoc development disadvantages

- Some important actions (testing, design) may go ignored
- Unclear when to start or stop each task
- Scales poorly to multiple people
- Hard to review or evaluate one's work

The later a problem is found in software,
the more costly it is to fix.

What makes a lifecycle?

- Requirements
- Design
- Implementation
- Testing
- Maintenance

How do we combine them?

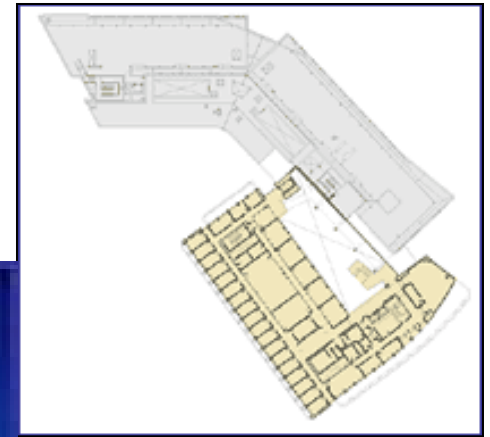
Benefits of using a lifecycle

- provides a work structure
- forces thinking about the “big picture”
- helps prevent decisions that are individually on target but collectively misdirected
- assists management and progress control

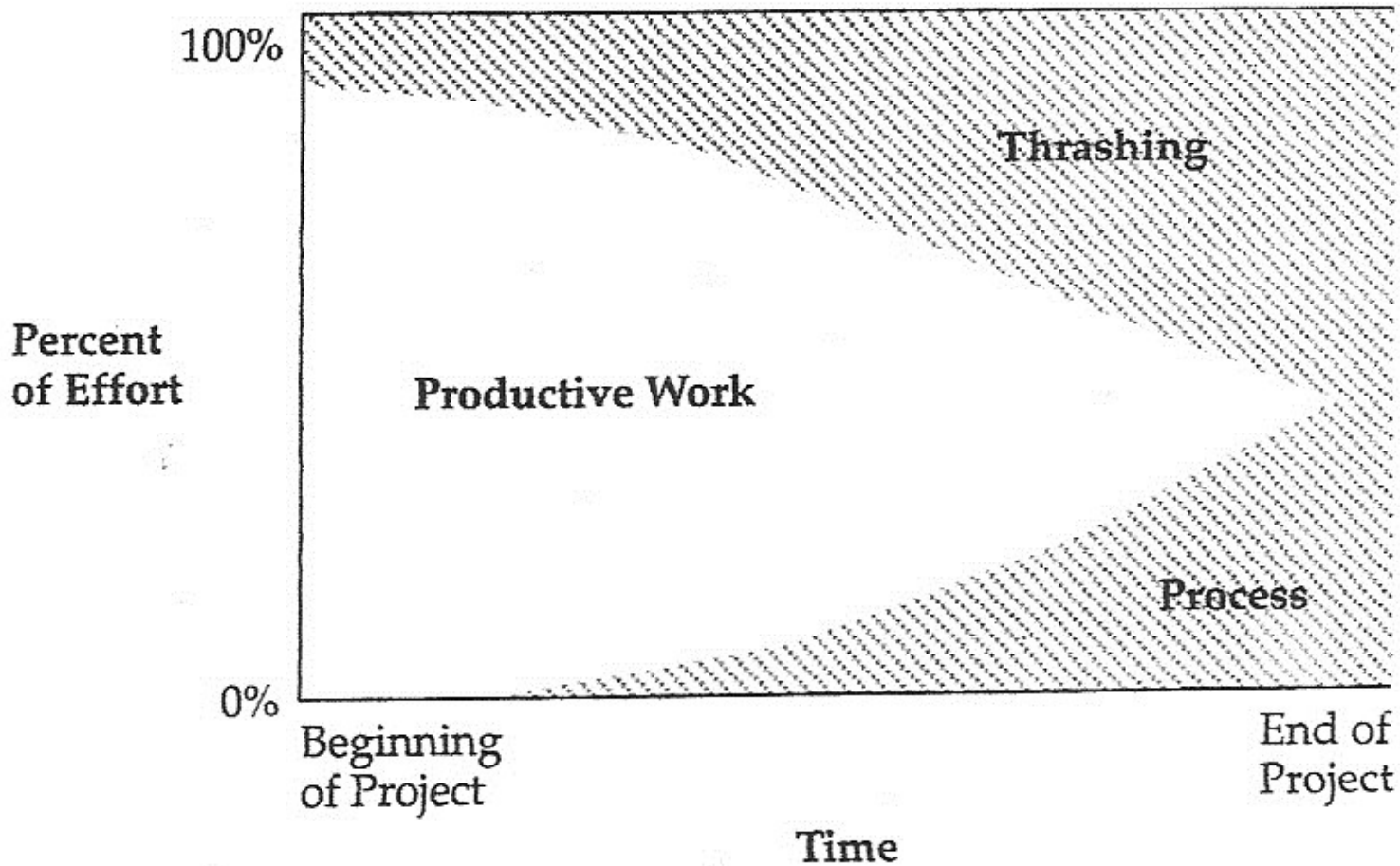
What are some drawbacks?

Are there analogies outside of SE?

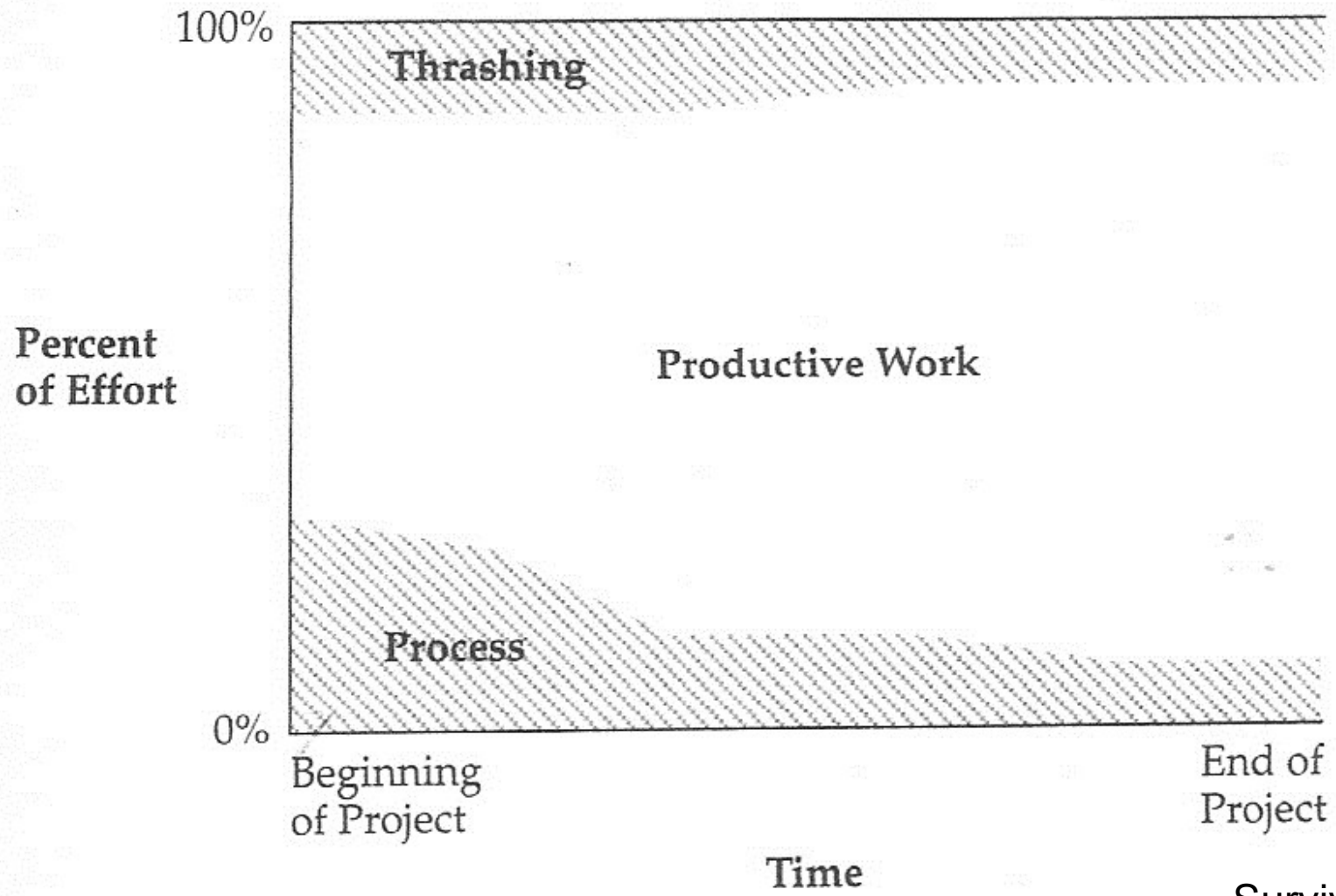
Consider the process of building the Paul Allen Center



Project with little attention to process



Project with early attention to process



Let's talk about some lifecycle models

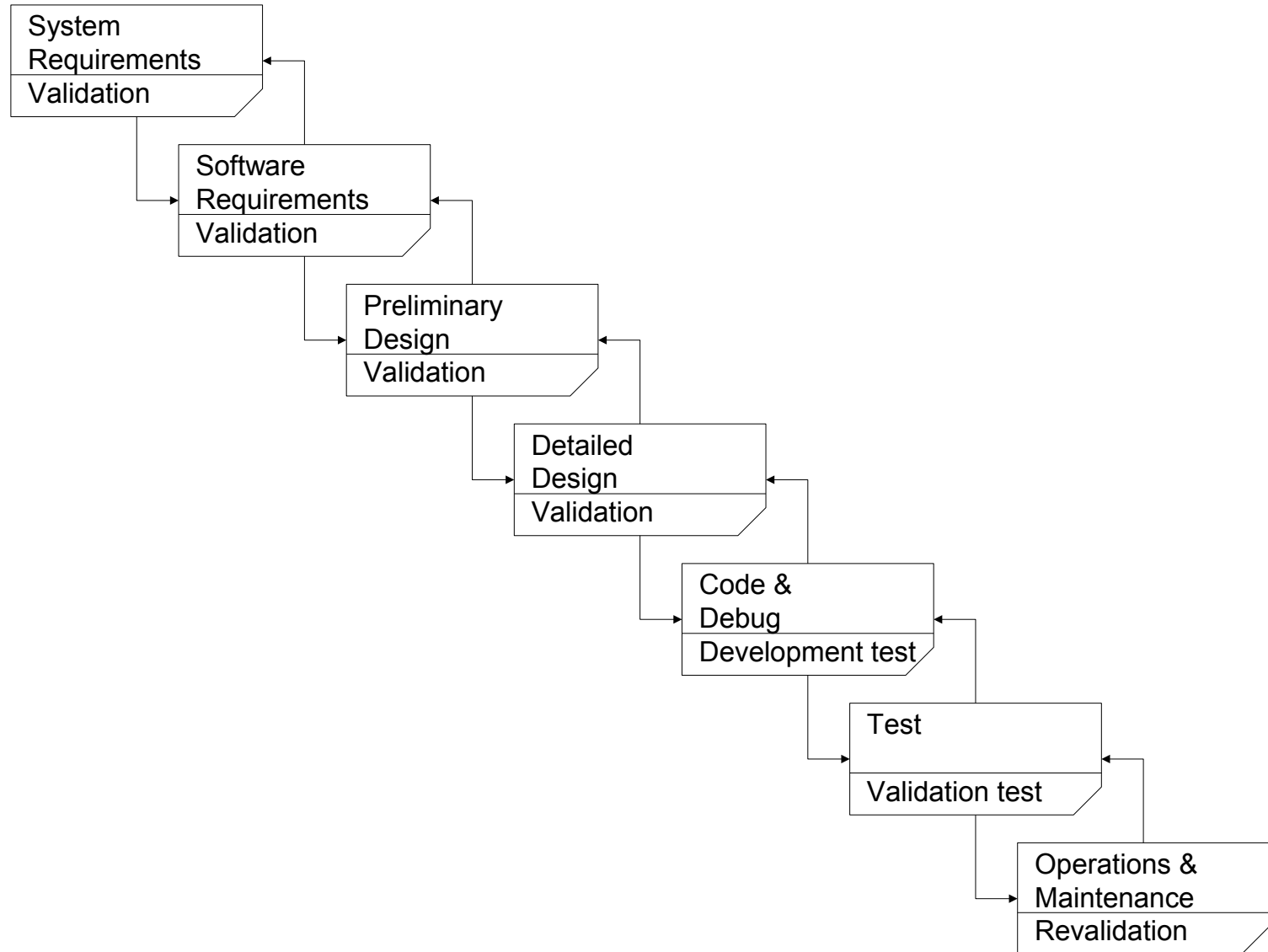
Code-and-fix model



Code-and-fix model

- Advantages
 - Low overhead
 - Applicable to small, short-lived projects
- Dangers
 - No way to assess progress and manage risks
 - Hard to accommodate changes
 - Unclear what and when will be delivered
 - Hard to assess quality

Waterfall model



Waterfall model advantages

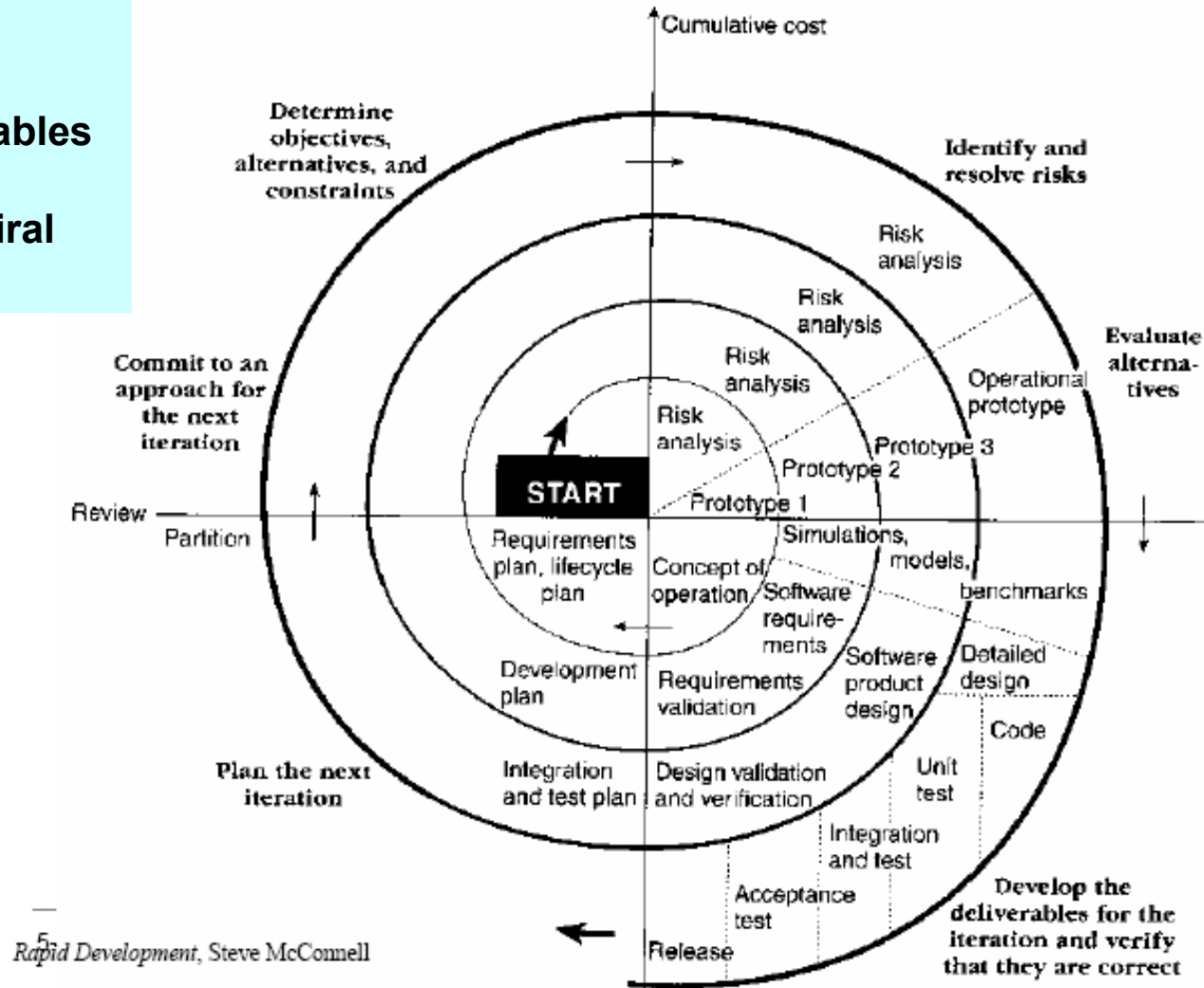
- Works well for well-understood projects
 - tackles all planning upfront
 - no midstream changes leads to efficient software development process
- Supports experienced teams
 - Orderly, easy-to-follow sequential model
 - Reviews help determine readiness to advance

Waterfall model limitations

- Difficult to do all planning upfront
- No sense of progress until the end
- Integration occurs at the very end
 - Defies the “integrate early and often” rule
 - Without feedback, solutions are inflexible
 - Final product may not match customer’s needs
- Phase reviews are massive affairs
 - It takes a lot of inertia and \$ to make changes

Spiral Model

- Determine objectives
- Identify and resolve risks
- Evaluate alternatives
- Develop and verify deliverables
- Plan next spiral
- Commit (or not) to next spiral



Spiral model

- Oriented towards phased reduction of risk
- Take on the big risks early
 - are we building the right product?
 - do we have customers for this product?
 - is it possible to use existing technology?
 - tomorrow's technology?
- Progresses carefully toward a result

Spiral model advantages

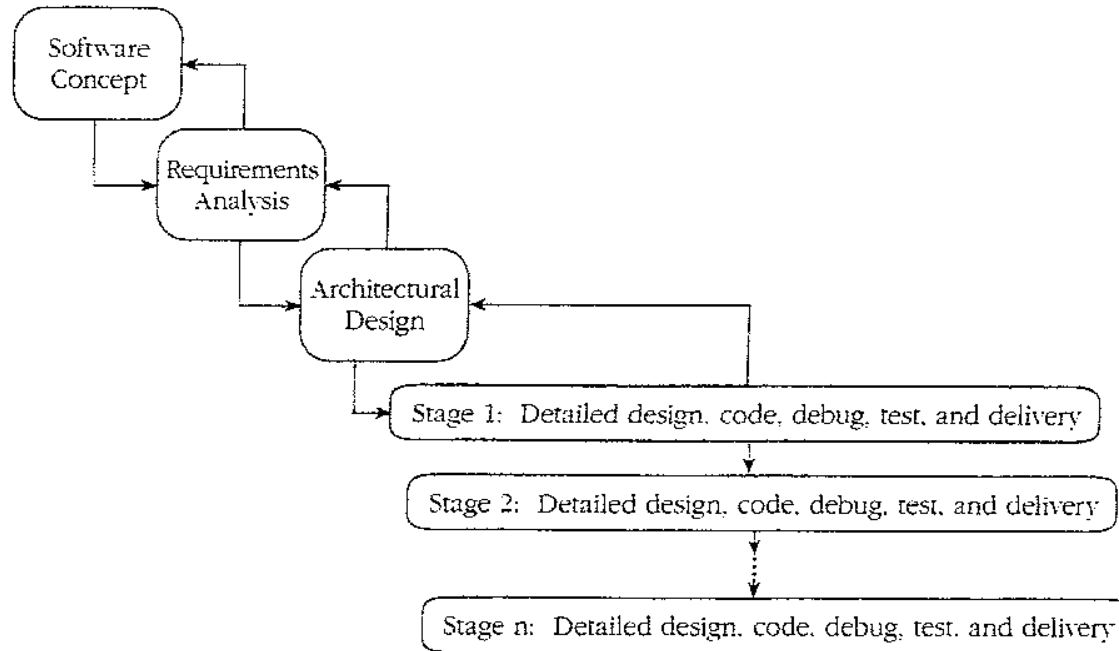
- Especially appropriate at the beginning of the project, allowing requirement fluidity
- Provides early indication of unforeseen problems
- Allows for change
- As costs increase, risks decrease!

Addresses the biggest risk first

Spiral model disadvantages

- A lot of planning and management
- Requires customer and contract flexibility
- Developers must be able to assess risk

Staged delivery model



first, waterfall-like

then, short release cycles: plan, design, execute, test, release
with delivery possible at the end of any cycle

Staged delivery model advantages

- Can ship at the end of any release cycle
- Intermediate deliveries show progress, satisfy customers, and lead to feedback
- Problems are visible early (e.g., integration)
- Facilitates shorter, more predictable release cycles

Very practical, widely used and successful

Staged delivery model disadvantages

- Requires tight coordination with documentation, management, marketing
- Product must be decomposable
- Extra releases cause overhead

What's the best model?

Consider

- The task at hand
- Risk management
- Quality / cost control
- Predictability
- Visibility of progress
- Customer involvement and feedback

Aim for good, fast, and cheap.
But you can't have all three at the same time.