Software Development Lifecycle

The Power of Process

Readings

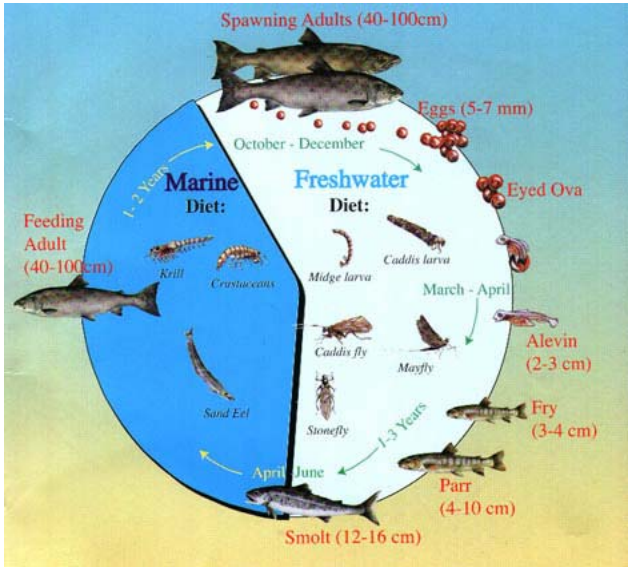
- "*Rapid Development*", Steve McConnell
  - Chapters 7, 10, 21, 25, 35, 36
- "*Anchoring the Software Process*", Barry Boehm
  - Pages 1-10 in particular

Outline

- What is a software development lifecycle?
- Why do we need a lifecycle process?
- Lifecycle models and their tradeoffs
  - "Code-and-fix"
  - Waterfall
  - Spiral
  - Evolutionary prototyping
  - Staged delivery
- Main recurring themes

What do we mean by a lifecycle?

Over to you … what do you think?

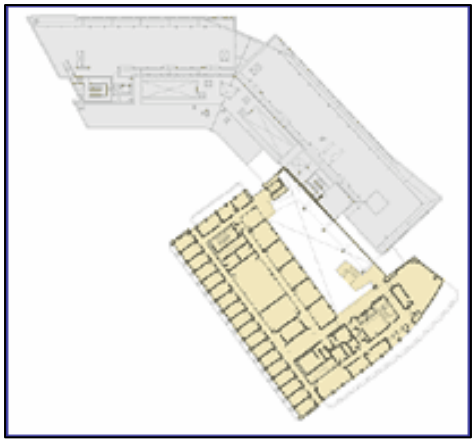


- The main function of a lifecycle model is to establish order in which project events occur from project conception to project end-of-life

- Typical events include
  - Specification, design, implementation, test, release
  - But they usually don't happen in nice clean little stages like this
  - So we develop various models to try to maintain the benefits and still be realistic

Are there analogies outside of SE?

- Consider the process of building the Paul Allen Center



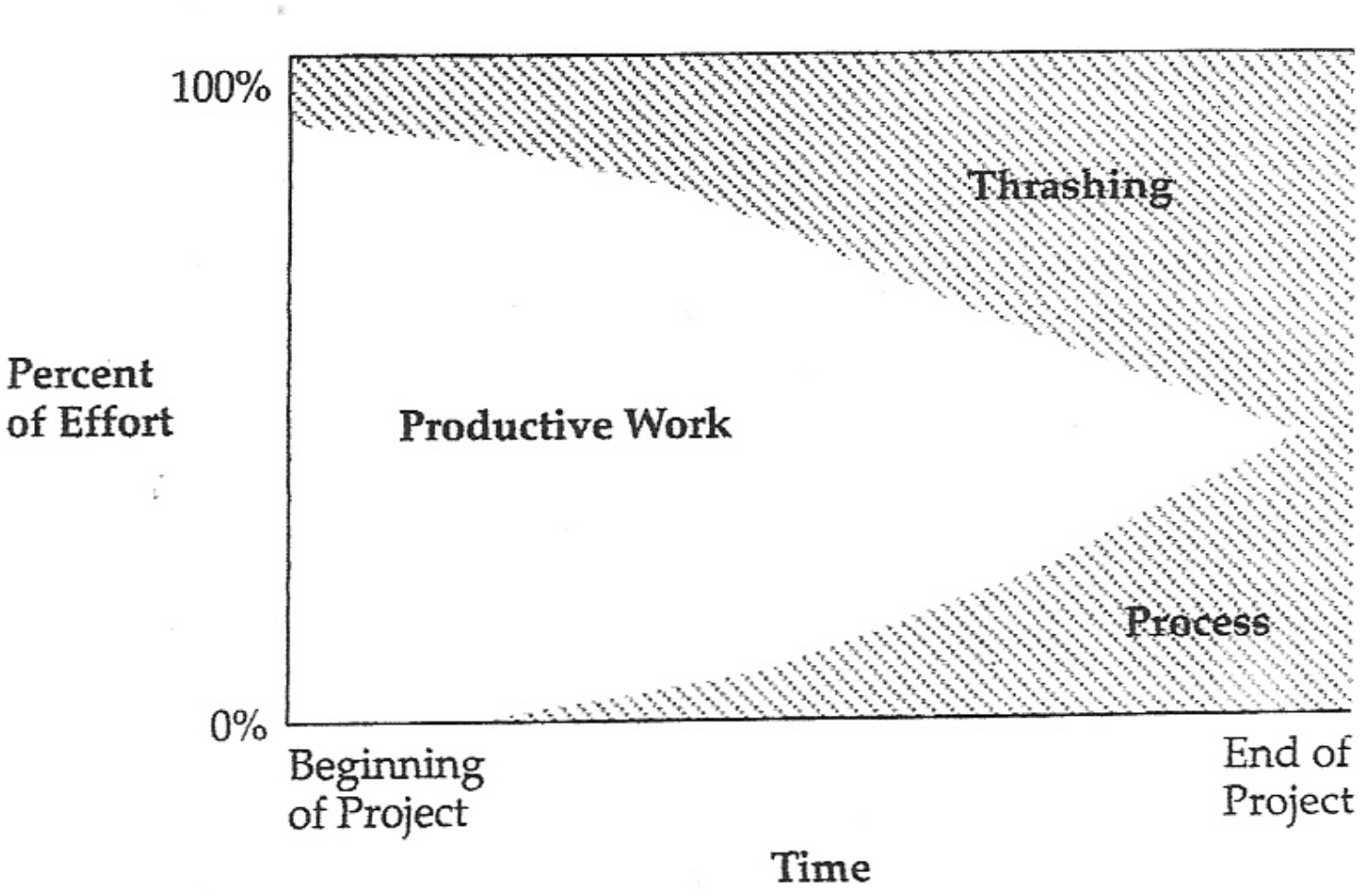
Is a lifecycle process really necessary?

I say “yes”, what about you? Why?

- It provides us with a structure in which to work
- It forces us to think of the “big picture” and follow steps so that we reach it without glaring deficiencies
- Without it you may make decisions that are individually on target but collectively misdirected
- It is a management tool, but not only for managers!

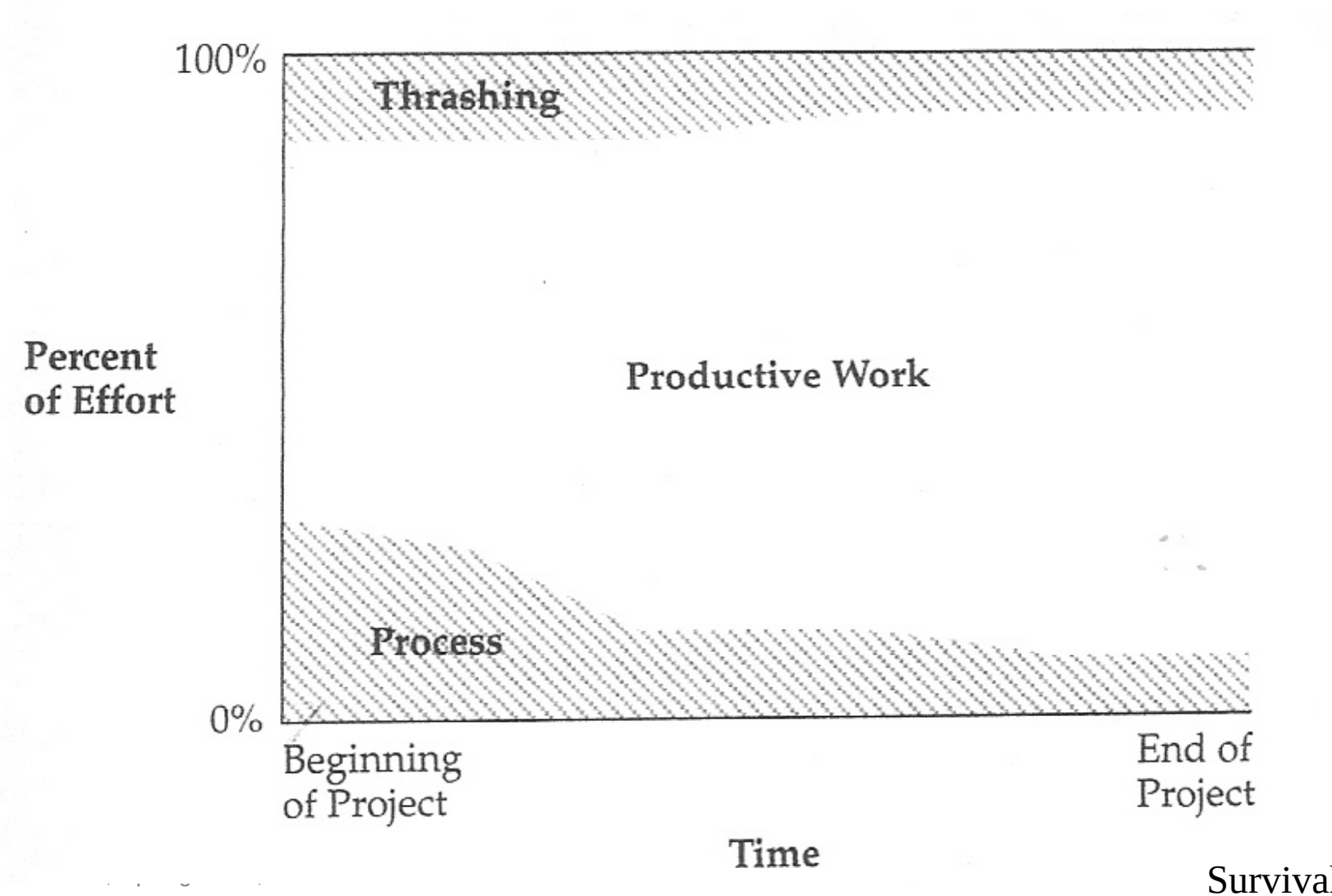
Do all projects need to follow a lifecycle process?

Project with little attention on process



Survival Guide:
McConnell

Project with early attention on process



Survival Guide:
McConnell

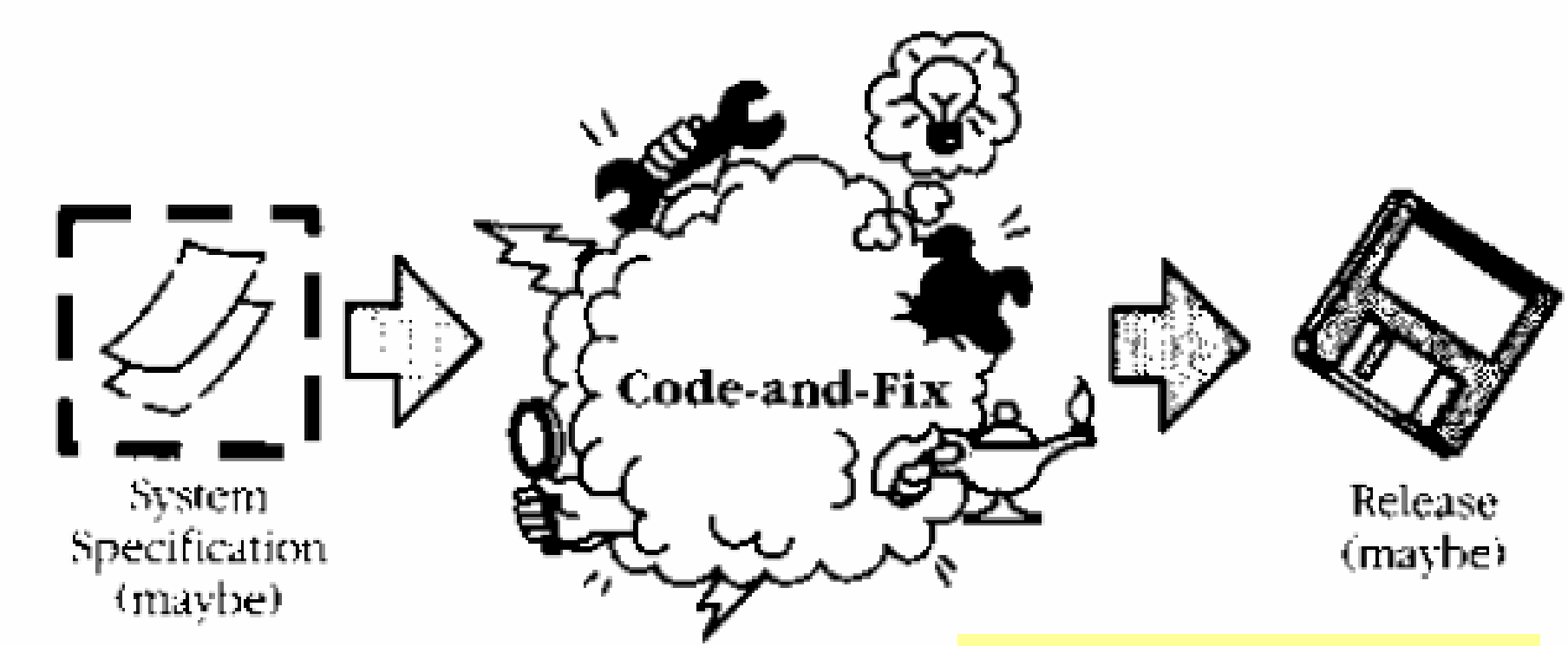
Onto the models…

These are fairly well known and used:

- "Code-and-fix"
- Waterfall
- Spiral
- Evolutionary prototyping
- Staged delivery

But there are many others (design-to-schedule, evolutionary delivery, variations on the above…)!

"Code-and-fix" Model



Can you think of a project you've developed this way?

"Code-and-fix" Model

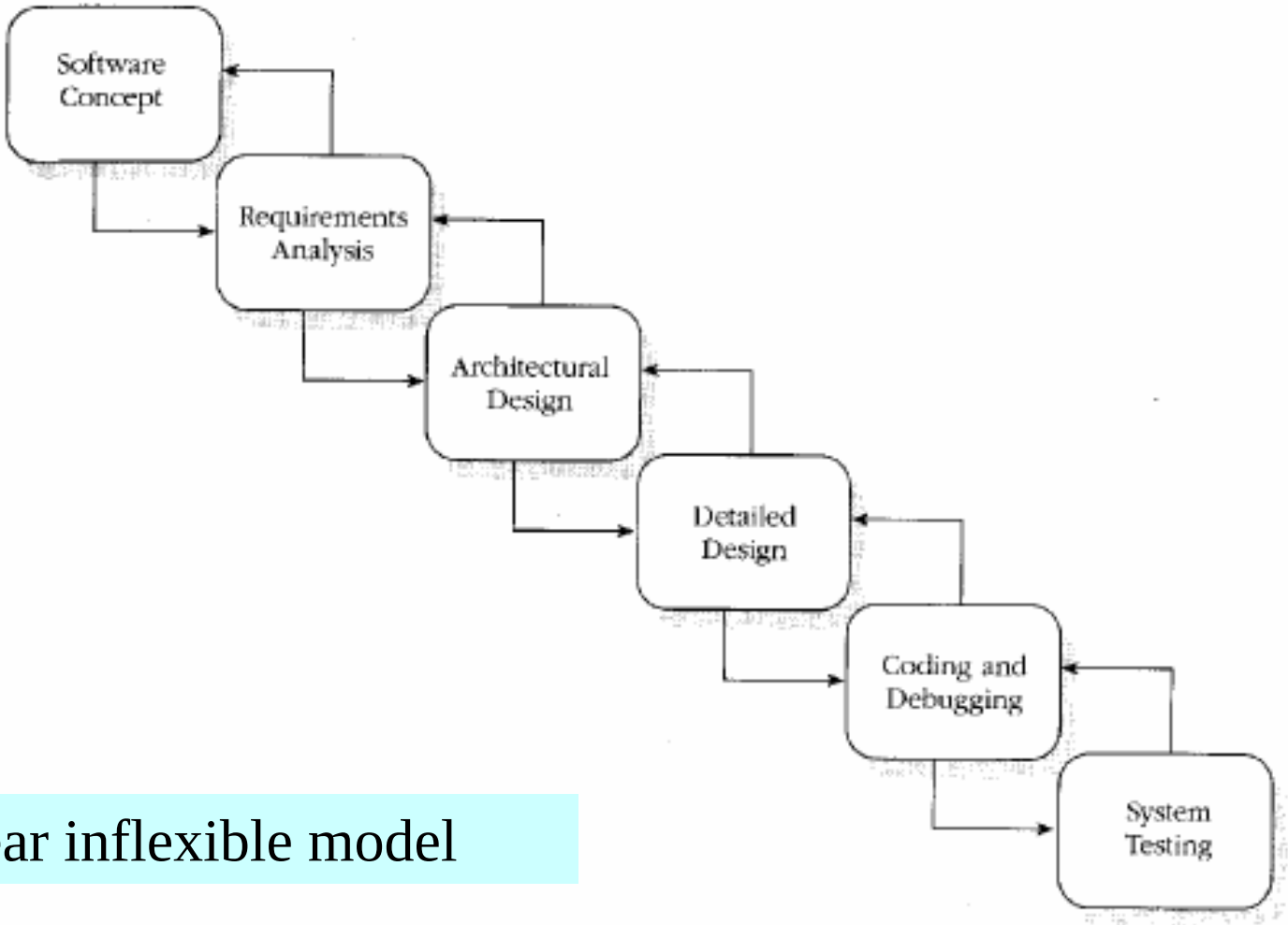
Advantages

- Little or no overhead - just dive in and develop, and see progress quickly
- Applicable *sometimes* for very small projects and short-lived prototypes

But

- Dangerous for most projects Why?
  - No way to assess progress, quality or risks
  - Unlikely to accommodate changes without a major design overhaul
  - Unclear delivery features (scope), timing, and support

Classic Waterfall Model



Linear inflexible model

Classic Waterfall Advantages

- Can work well for projects very well understood but complex
  - Tackles all planning upfront
  - The ideal of no midstream changes equates to an efficient software development process

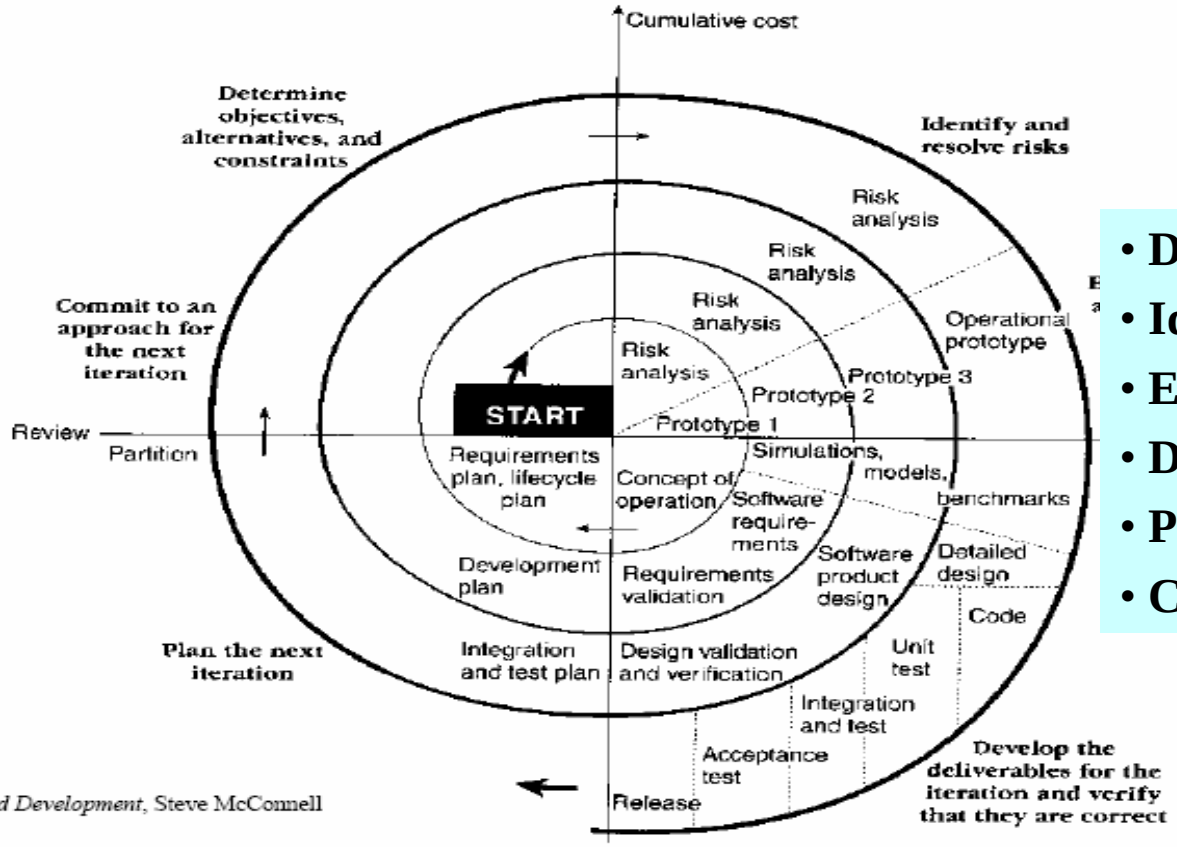
- Can provide support for an inexperienced team
  - Orderly sequential model that is easy to follow
  - Reviews at each stage determine if the product is ready to advance

Classic Waterfall Limitations

Your turn …

- Difficult to specify all reqs of a stage completely and correctly upfront
  - completely → lots and lots of detail
  - correctly → every single detail is correct
- No sense of progress until the very end
  - "so far so good"
  - Nothing to show to anxious customers ("we're 90% done")
- Integration occurs at the very end
  - Definite setup for failure -integrate early and often is the rule in practice
  - Solutions are inflexible, no allowance for feedback of into discovered later
  - Inasmuch, what is delivered may not match customer real needs
- Phase reviews are massive affairs
  - It takes a lot of inertia ($$) to make any change given the material behind the current path

Spiral Model – Risk Oriented



Rapid Development, Steve McConnell

- **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 and make some decisions
  - are we building the right product?
  - do we have any customers for this product?
  - is it possible to implement the product with the technology that exists today? tomorrow?
- Walks carefully to a result (tasks can be more clear each spiral)

Can you think of a project that could benefit from this model?

Spiral Model

Advantages

- Especially appropriate at the beginning of the project when the requirements are still fluid
- Provides early indication of unforeseen problems
  - Checkpoints at the end of each spiral, based on greatest risks
- As costs increase, risks decrease!
  - Always addresses the biggest risk first

Limitations

- Requires a level of planning and management (cost)