COSC 310: Software Engineering

Dr. Bowen Hui

University of British Columbia Okanagan

Admin

- A2 is up
  - Don't forget to keep doing peer evaluations
  - Deadline can be extended but shortens A3 timeframe
- Labs
  - This week: set up team repo
  - Next week: practice choosing SDLC

Software Process

- Is a set of activities that produce software product
- A structured way of carrying out activities
- E.g.,
  - Course registration process
  - Course credit transfer process
  - Online payment process

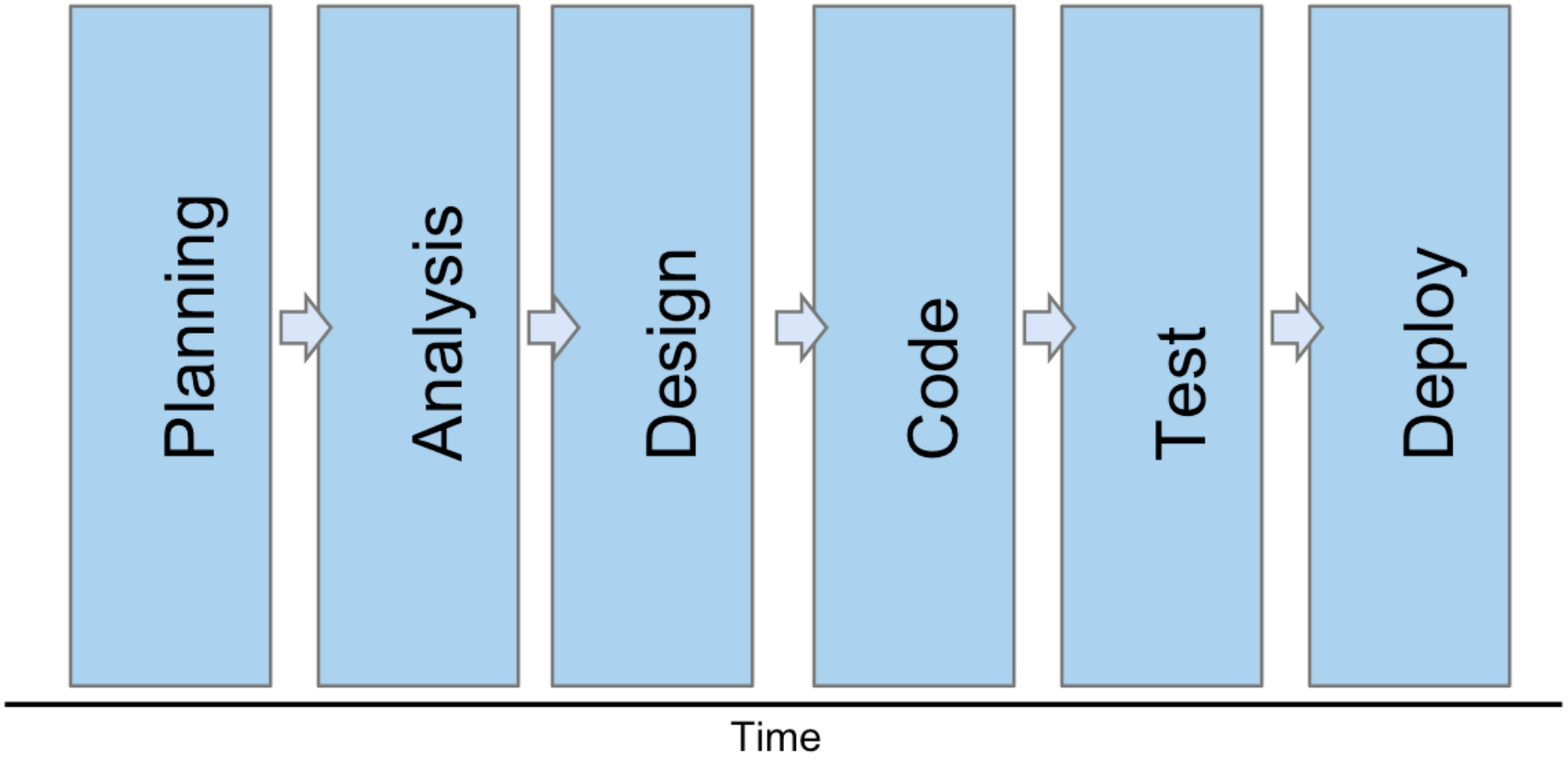
Fundamental Software Activities

- Planning – resource allocation and management
- Requirements Specification – identifies necessary features
- Design – architecture, modules, interfaces
- Development – implementation
- Testing – validation of correctness
- Documentation – describes product
- Maintenance – ongoing error correction
- Evolution – ongoing requirements changes

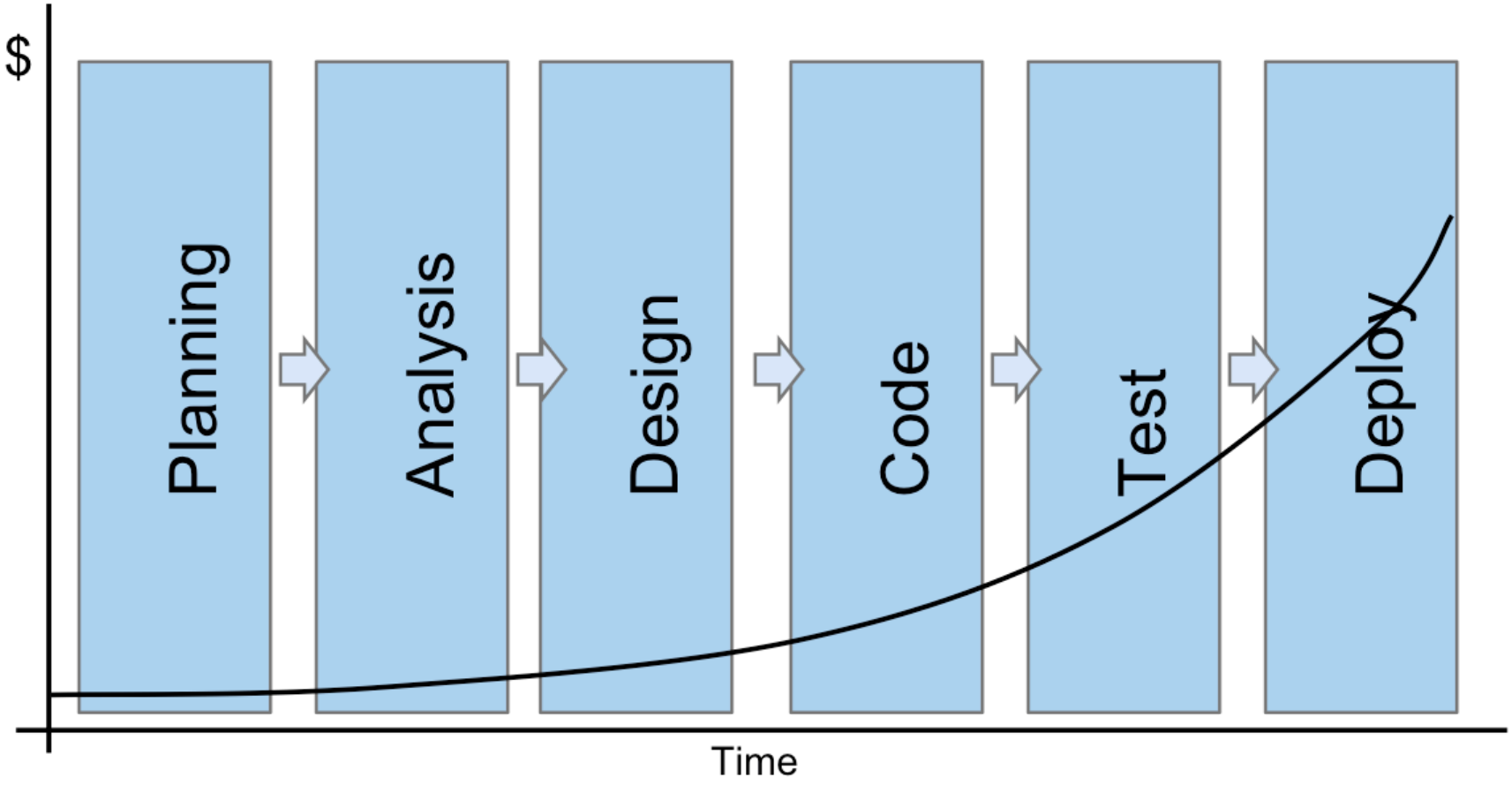
Goals of Software Activities

- Clarify steps to be performed
- Produce tangible work products
- Enable others to review work products
- Specify next steps

Typical Stages of Software Development



Software Development Cost



Scenario 1

- Assume:
  - Cost of maintenance is 75% of total cost
  - Cost of development is 25% of total cost

- If development cost is $250K, what is the maintenance cost?

Scenario 2

- Client asks for accounting information system to be built in order to support 20 users
  - Users all need new computers
  - Client requires software to be completed in 4 weeks
  - Budget is $50K for hardware and software

- Is this a good deal?

Scenario 3

- Relative cost of fixing errors at various stages are:
  - Specification (3)
  - Design (5)
  - Implementation (50)
  - Maintenance – after deploy (300)
- If cost to find and fix an error in design is $100, what are the costs for other stages?

Simple Software Process (School work)

- Read question on assignment
- Think about it for a few minutes (design?)

Simple Software Process (School work)

- Read question on assignment
- Think about it for a few minutes (design?)
- Start coding – compile when done coding

Simple Software Process (School work)

- Read question on assignment
- Think about it for a few minutes (design?)
- Start coding – compile when done coding
- Compile – get 100+ errors
- Ditch assignment

Simple Software Process (School work)

- Read question on assignment
- Think about it for a few minutes (design?)
- Start coding – compile when done coding
- Compile – get 100+ errors
- Ditch assignment
- Resume coding next day… can't understand it so restart coding
- Compile – get 50+ errors
- Ditch assignment again

Simple Software Process (School work)

- Read question on assignment
- Think about it for a few minutes (design?)
- Start coding – compile when done coding
- Compile – get 100+ errors
- Ditch assignment
- Resume coding next day… can't understand it so restart coding
- Compile – get 50+ errors
- Ditch assignment again
- Night before due date: marathon coding
- Program "mostly" works – submit + never see it again

How to Improve Process?

How to Improve Process?

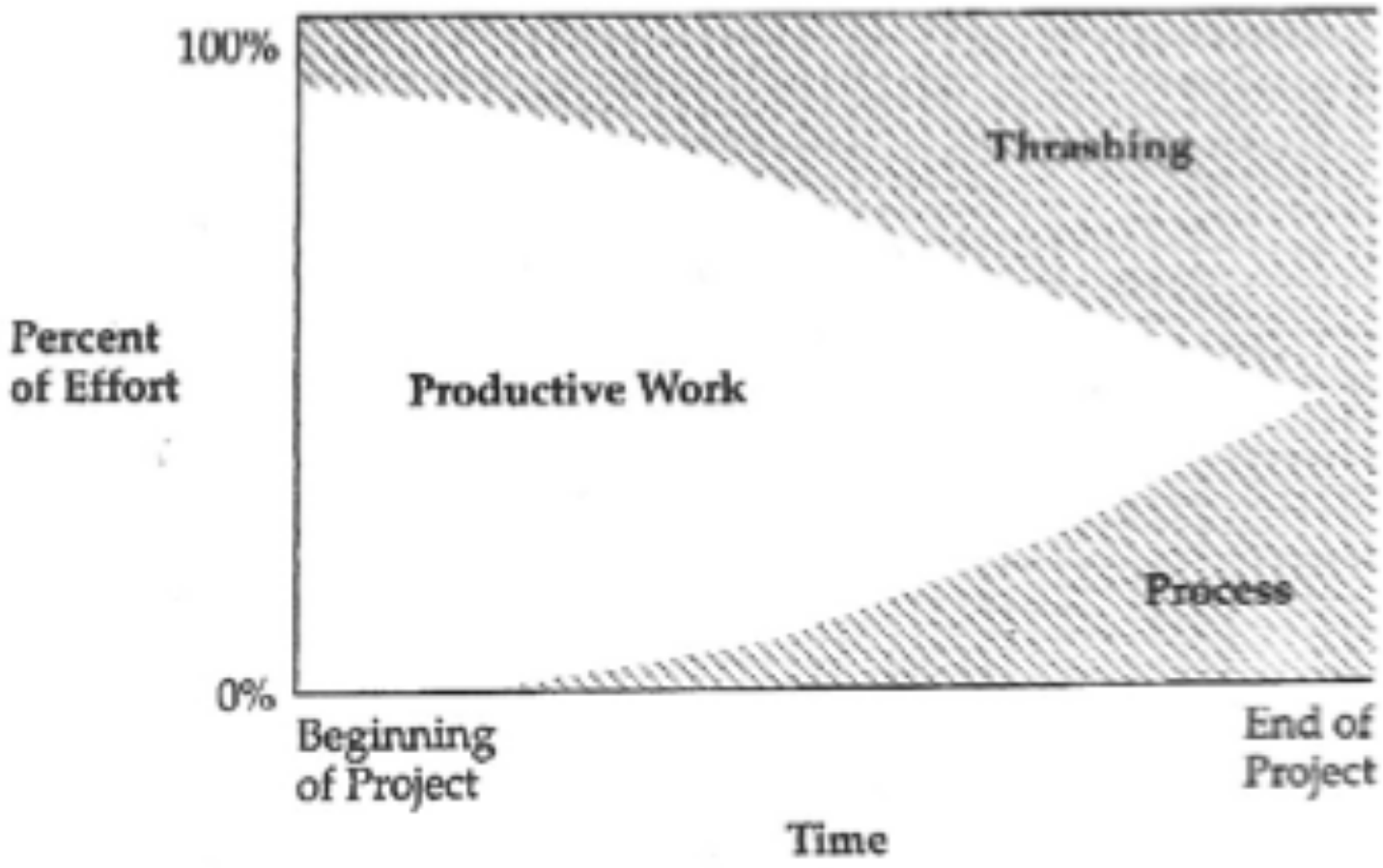
- Ask clarification questions if specs aren't clear
- Design different solutions, consider the best one
- Repeat: code + test a bit at a time
- Clean up code as you go

Managing Big Projects

- Complexity grows quickly for large projects
- Different teams working on different activities
  - Necessary to facilitate communication
- Need standardizations and careful management in order to stay within budget and time
- Helpful to use CASE tools to support large projects

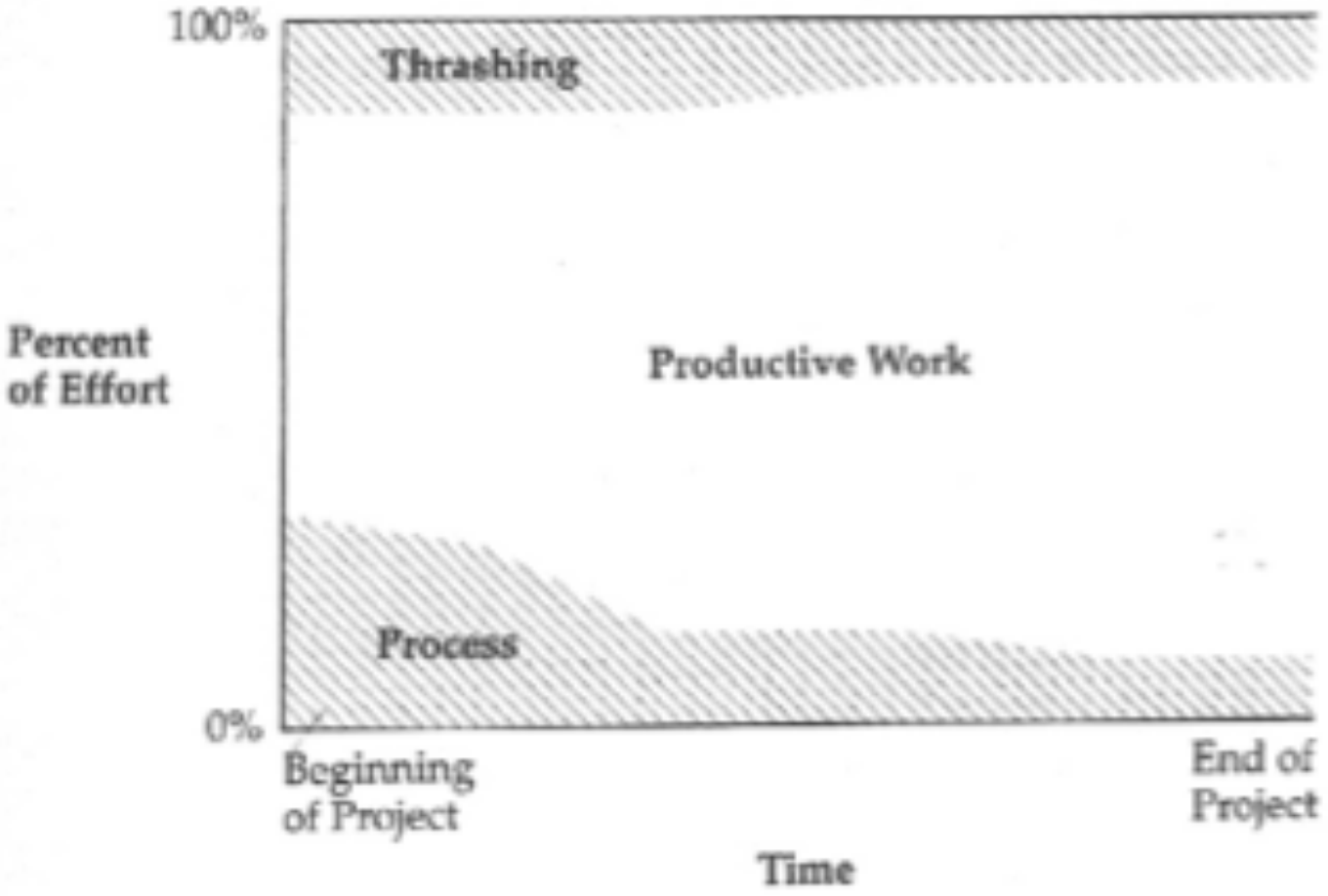
Productivity Work Analysis (McConnell)

- Little attention paid to process



Productivity Work Analysis (McConnell)

- Early attention paid to process



Why Should You Care?

- As a programmer:
  - Know the lingo
  - Understand the big picture
  - Understand your role
  - Feel less frustrated
- As a project leader:

Why Should You Care?

- As a programmer:
- As a project leader:
  - Assess risks
  - Choose appropriate process model
  - Have successful project

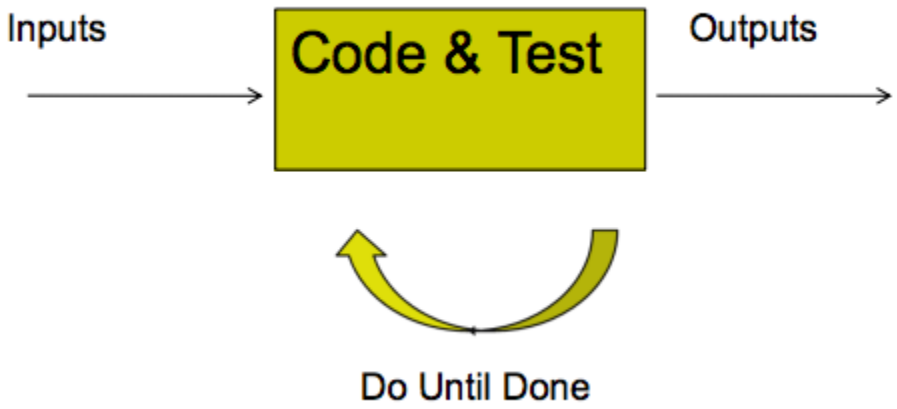
Process Management

- Software lifecycle (SDLC) describes the process of software development activities
- Discipline of defining, implementing, maintaining process in a project
- Adopted/created by PM
- Measures project progress
- Ensures correct execution of organization's procedures and policies

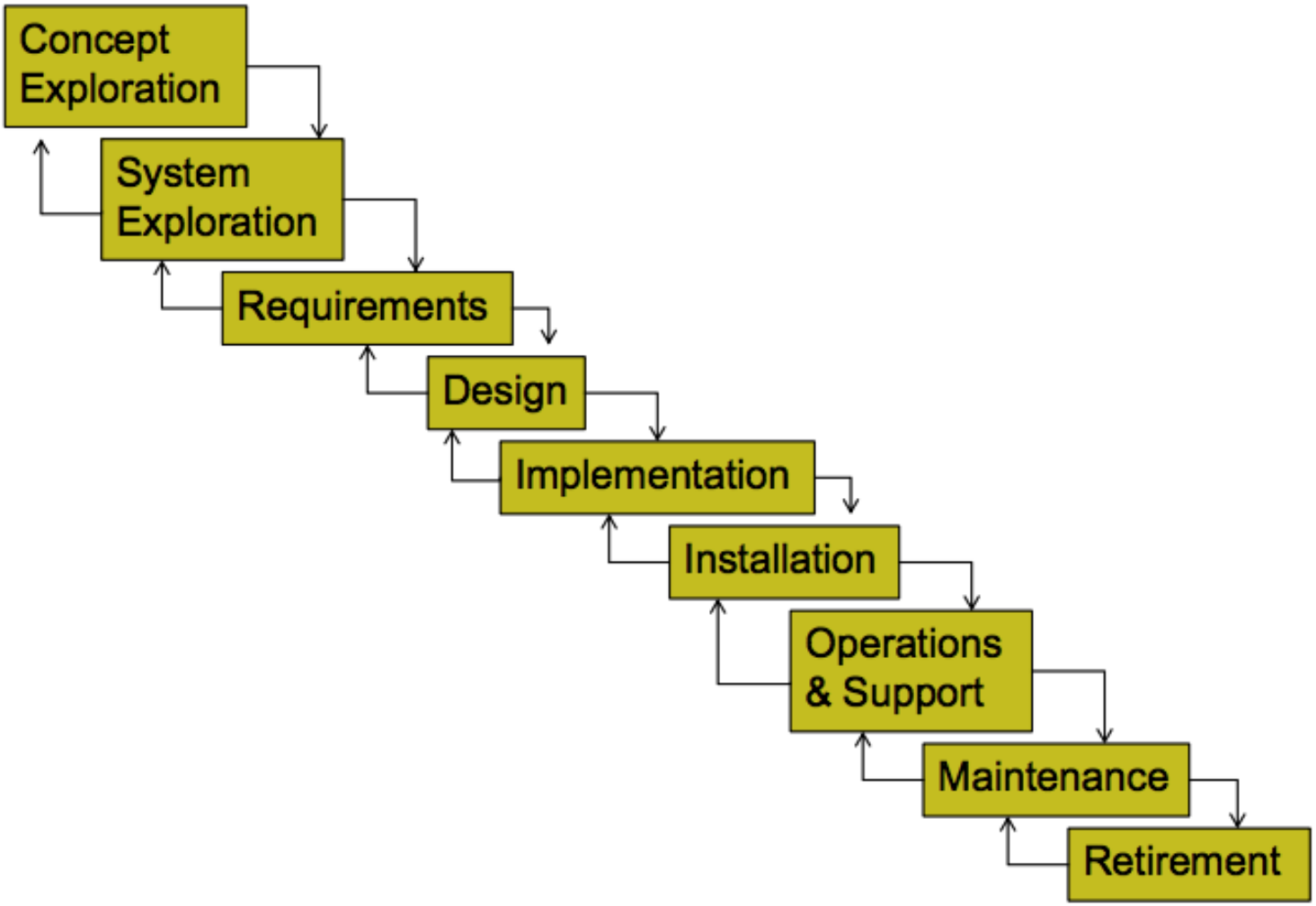
“Starter” SDLC Models

- Do-until-Done
- Waterfall
- V-Shaped
- Rapid Prototyping
  - Rapid application development (RAD)
  - Incremental
  - Spiral
- Agile Development
  - Scrum
  - XP

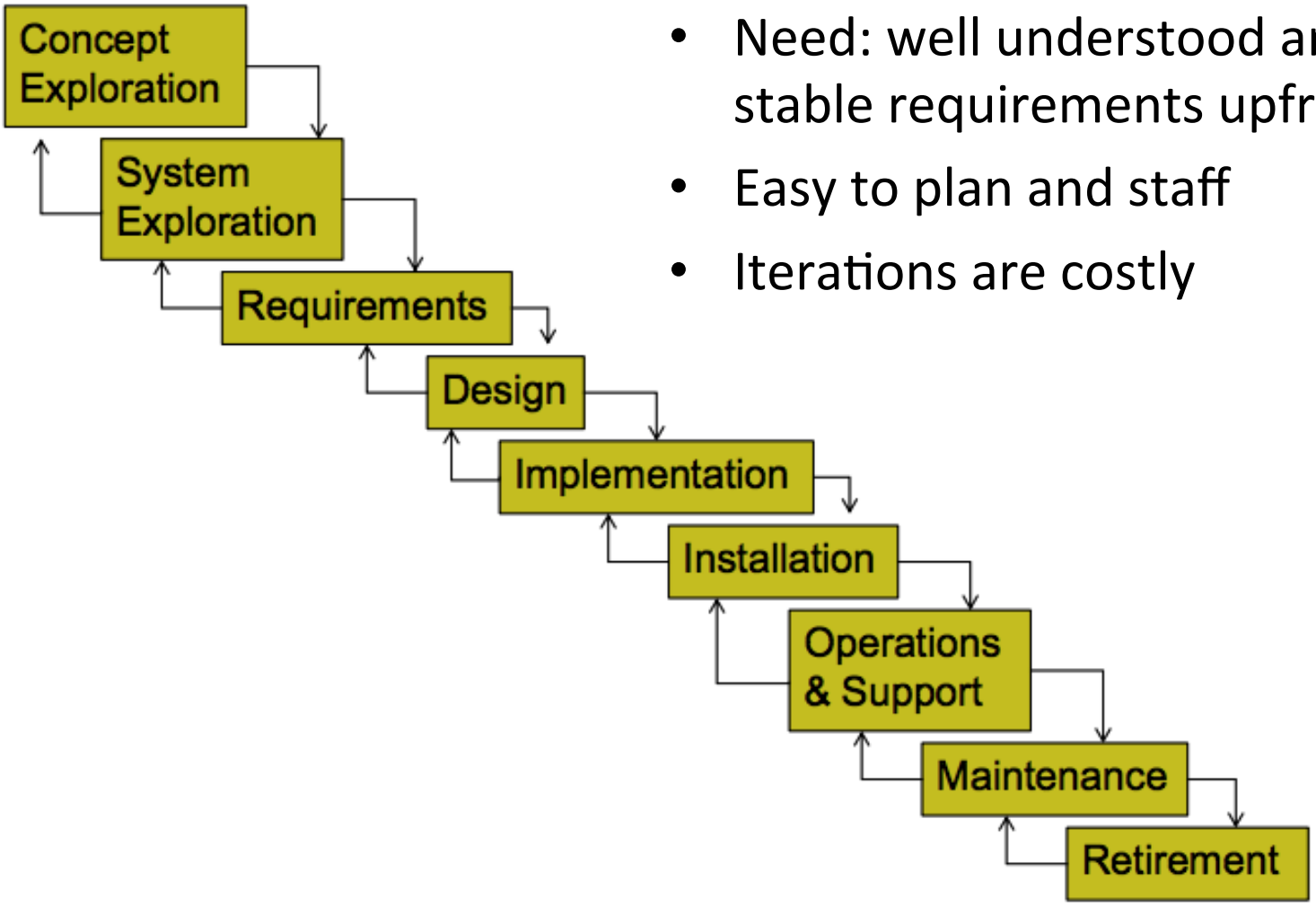
Do-Until-Done



Waterfall (several variants)

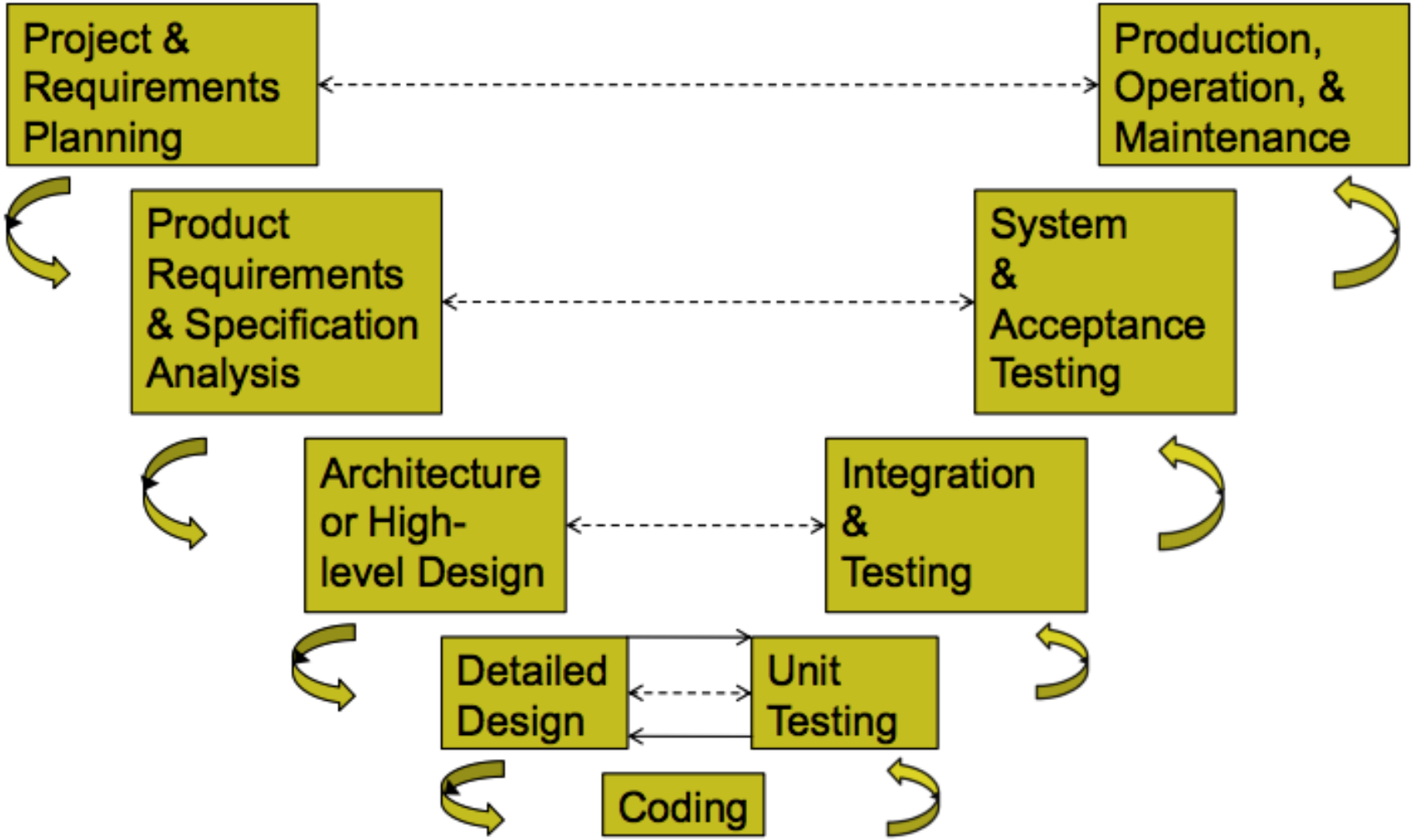


Waterfall (several variants)

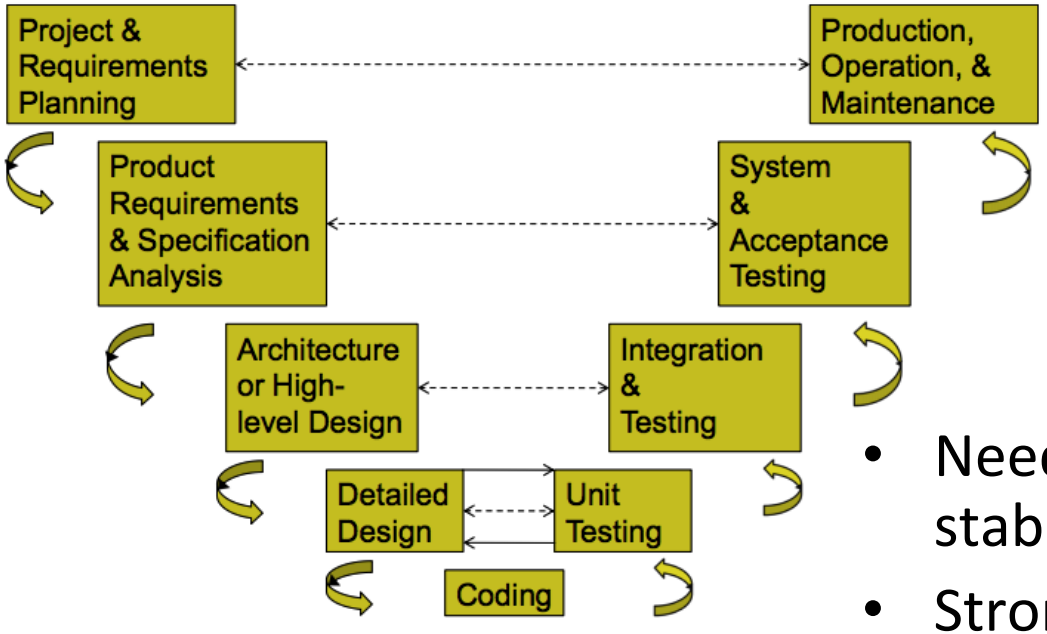


- Need: well understood and stable requirements upfront
- Easy to plan and staff
- Iterations are costly

V-Shaped


Project & Requirements Planning
Production, Operation, & Maintenance
Product Requirements & Specification Analysis
System & Acceptance Testing
Architecture or High-level Design
Integration & Testing
Detailed Design
Unit Testing
Coding

V-Shaped



- Need: well understood and stable requirements upfront
- Strong emphasis on verification and validation
- Good for systems requiring high reliability

Rapid Prototyping Models

- Single hardest part of building software is deciding what to build
- Challenge for team
  - Devise process that will discover, define, develop real requirements
- Central idea
  - Create a prototype of software that may require us to throw away at the next iteration
  - Prototype = easy to build, readily modifiable/extensible, partially specified working model of overall system
- Good approach if system is something *new*

Rapid Prototyping Model

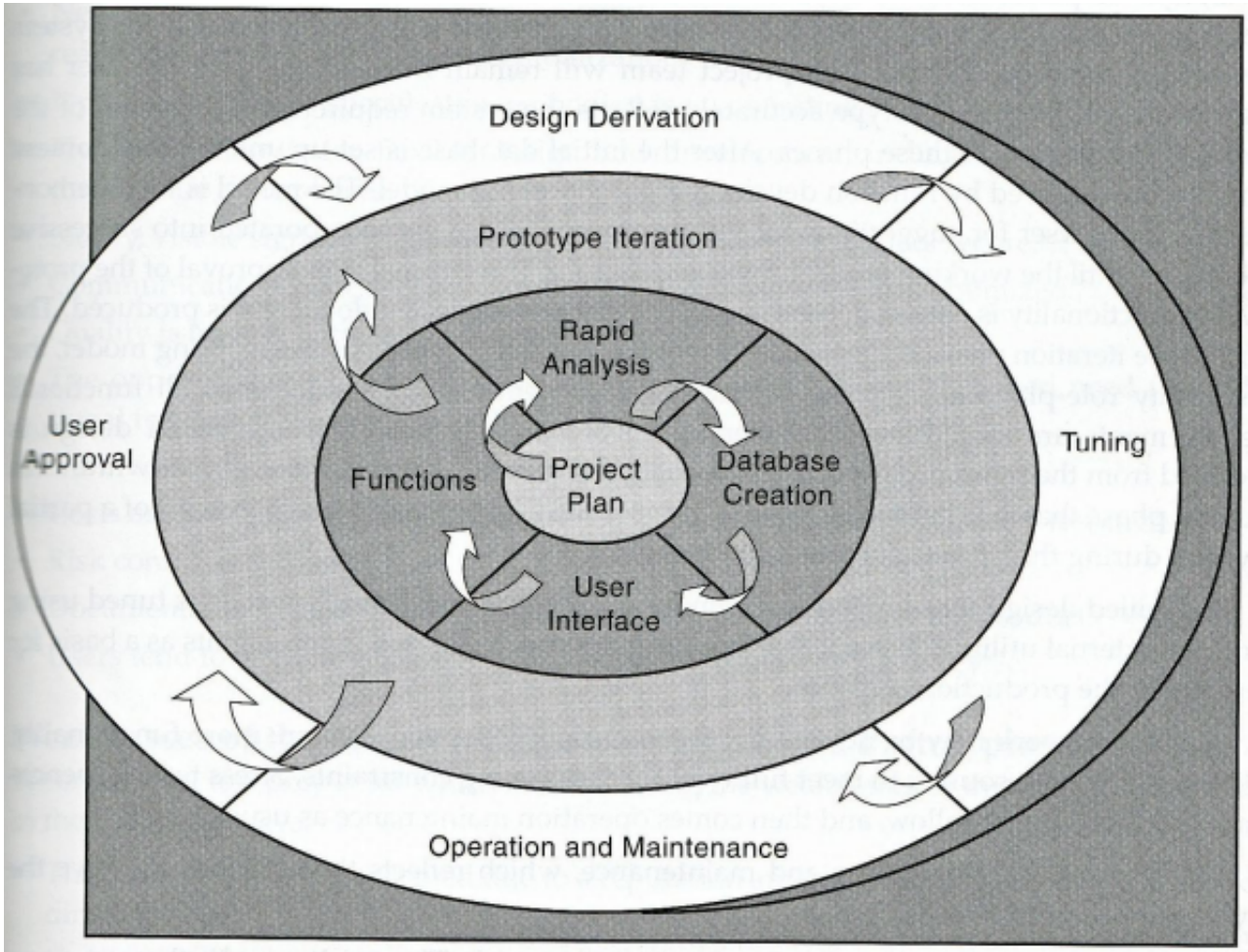


FIGURE 4–11
The Structured Evolutionary Rapid Prototyping Model

Just a Proof-of-Concept

- Prototype has limited functionality
  - Limited functionality
  - Limited platform interoperability
  - Need to manage client expectations
- Non-functional requirements not considered
  - Reliability? Performance? Usability?

When to use Rapid Prototyping?

- When requirements are ill-understood
  - Too complex
  - Always evolving requirements
- Medium or high risk projects
- When only proof-of-concept is needed
- Suitable for UI intensive systems and research projects
- Often used in combination with other SDLC

Three Types of Rapid Prototyping

- Rapid application development (RAD)
- Incremental
- Spiral

RAD

- Created by IBM 1980s
- Quick turnaround time: ~60 days
- Users involved in ALL phases
- Heavy dependence on:
  - Code generators
  - Screen generators
  - Other productivity tools
- More work for users in planning and design

RAD Model

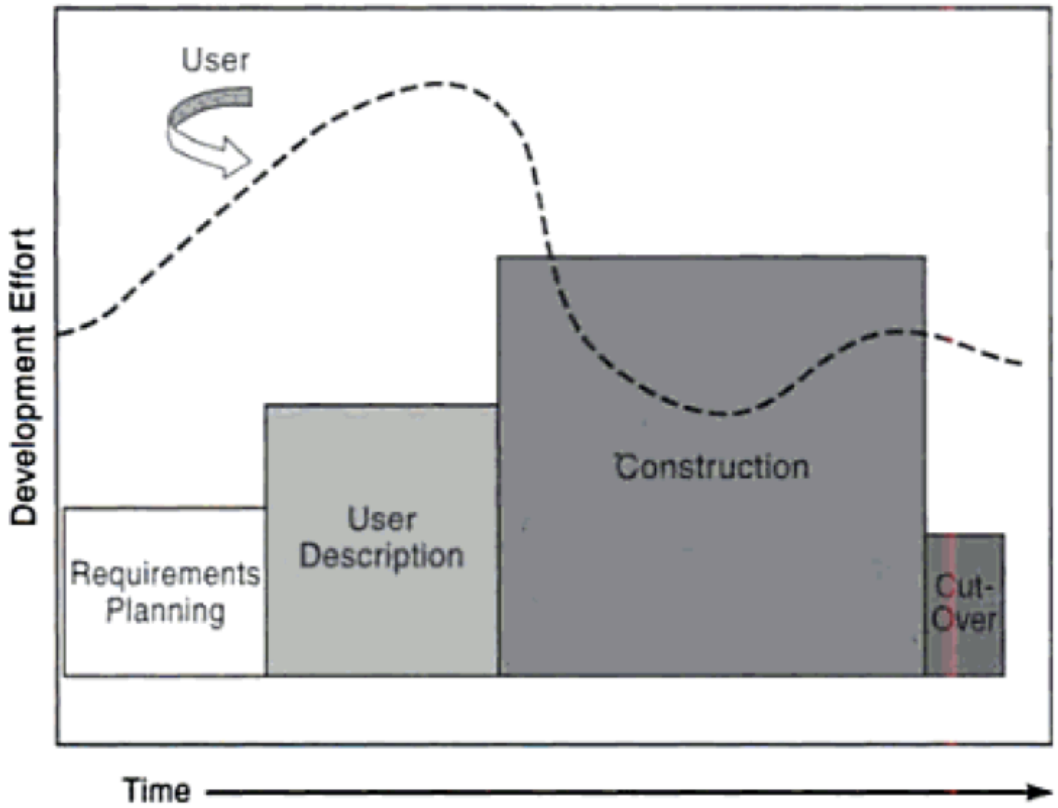


FIGURE 4–12
The Rapid Application Development Model

Phases:

- Joint Requirements Planning
- Joint Applications Development
- Code generation
- User acceptance

When to use RAD

- Requirements are well-understood
- Low risk projects where performance and reliability are not an issue
- Short development times
- High end-user involvement
- Automated tools available
- Ex: information system

Incremental

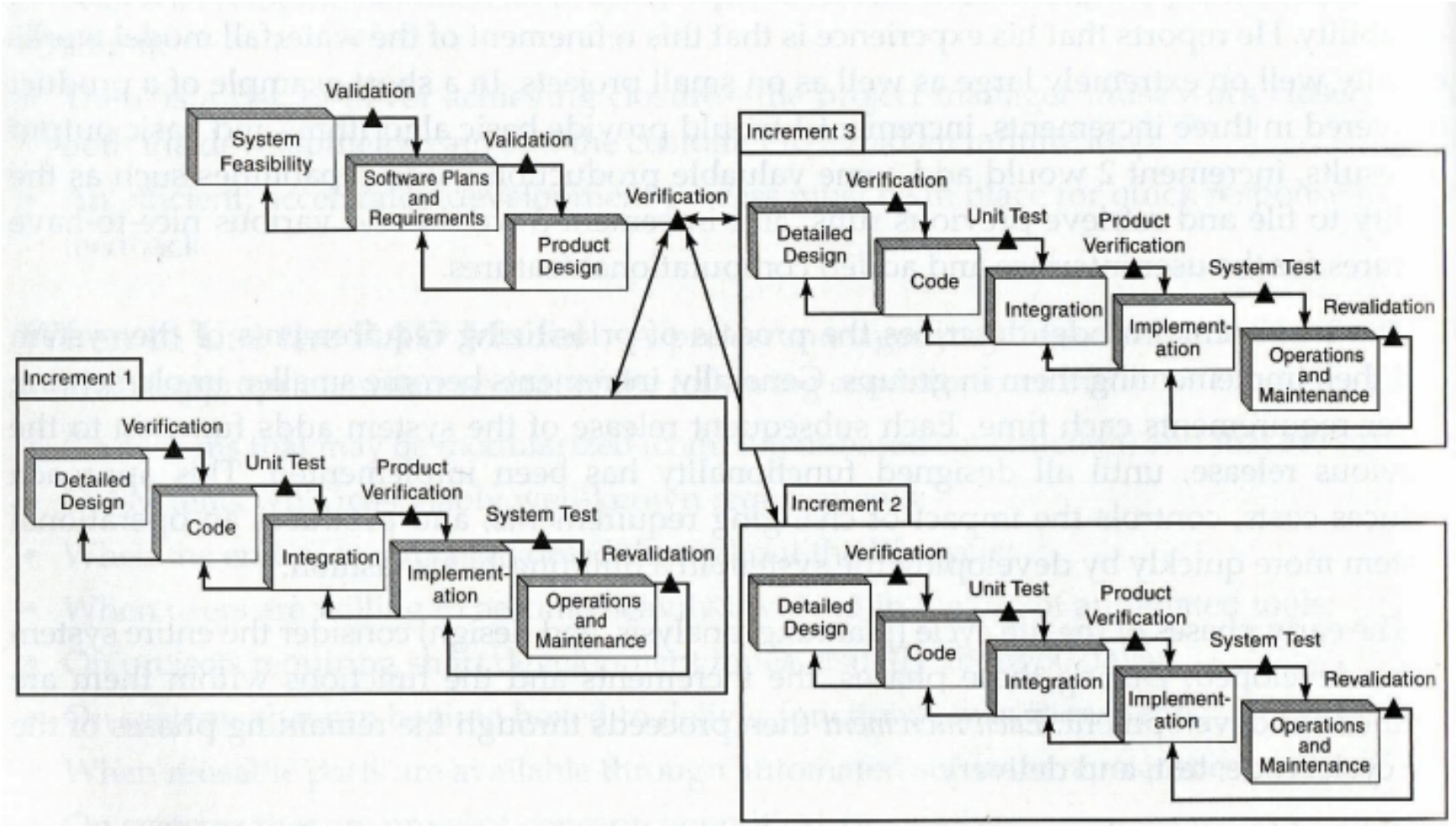


FIGURE 4–13
The Incremental Model

Main features of Incremental

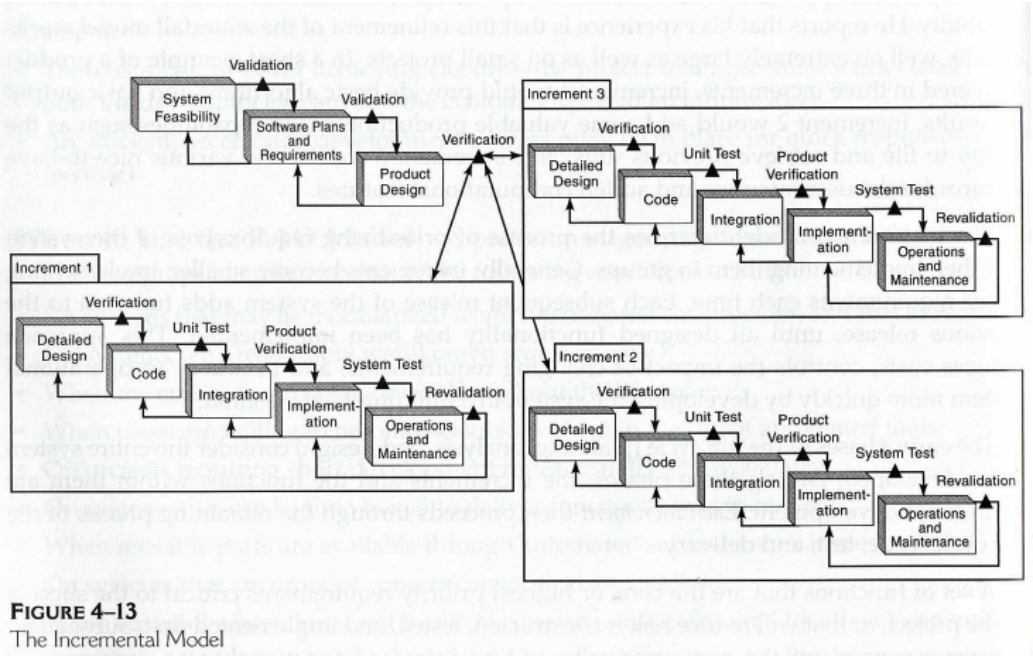


FIGURE 4–13
The Incremental Model

- Does not allow iterations
- Complete system definition required in order to create meaningful chunks
- Divide-and-conquer approach
- Prioritizes important functionality

Spiral

- Focuses on risk analysis and management
  - Cost
  - Life
  - High-profile missions
- Takes advantage of strengths from:
  - Waterfall
  - Prototyping
  - Incremental

Main features of Spiral

- Reviews (requirements, design, product)
- Specific deliverables
- Coding is de-emphasized

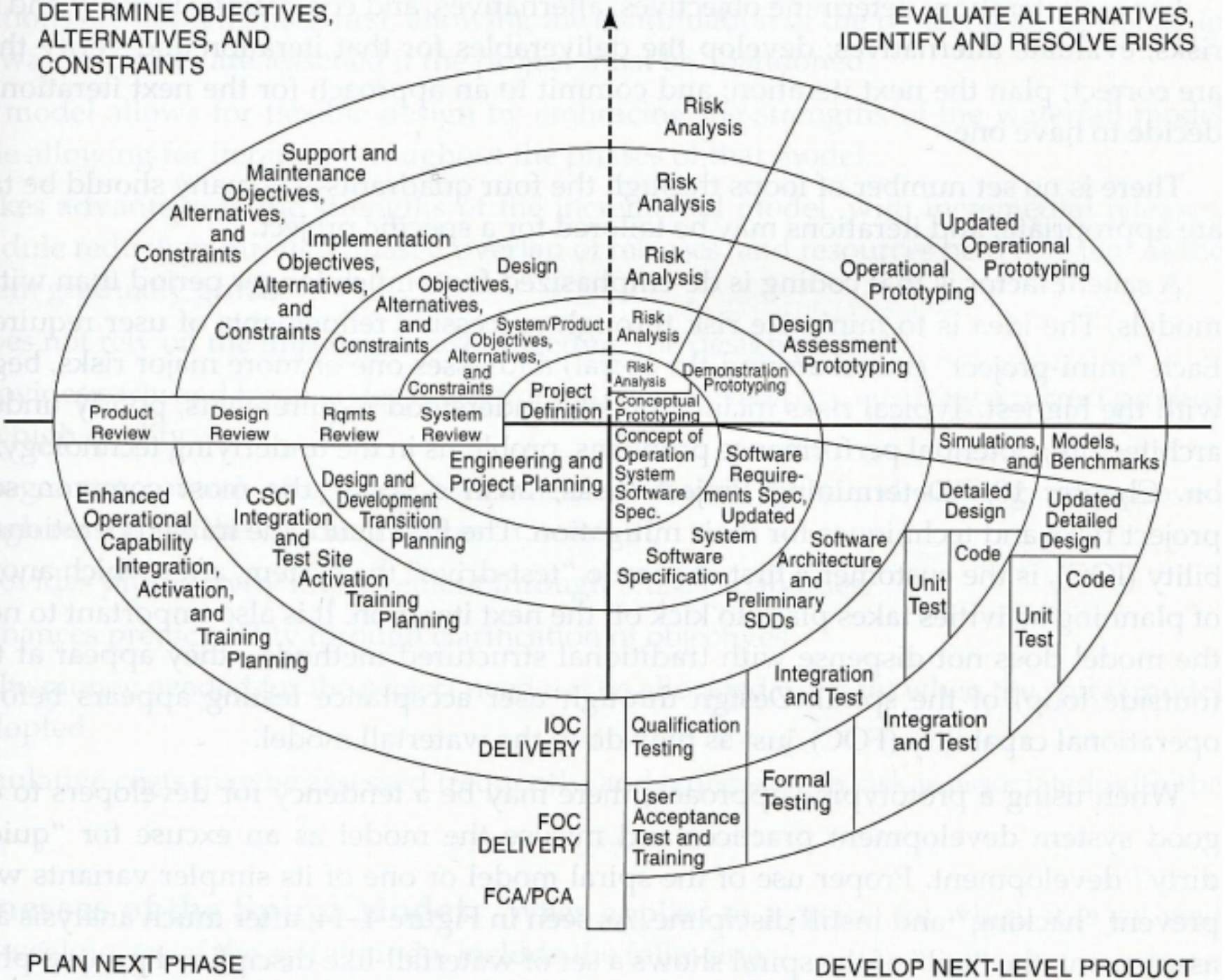
waterfall

- Allows users to see system early on

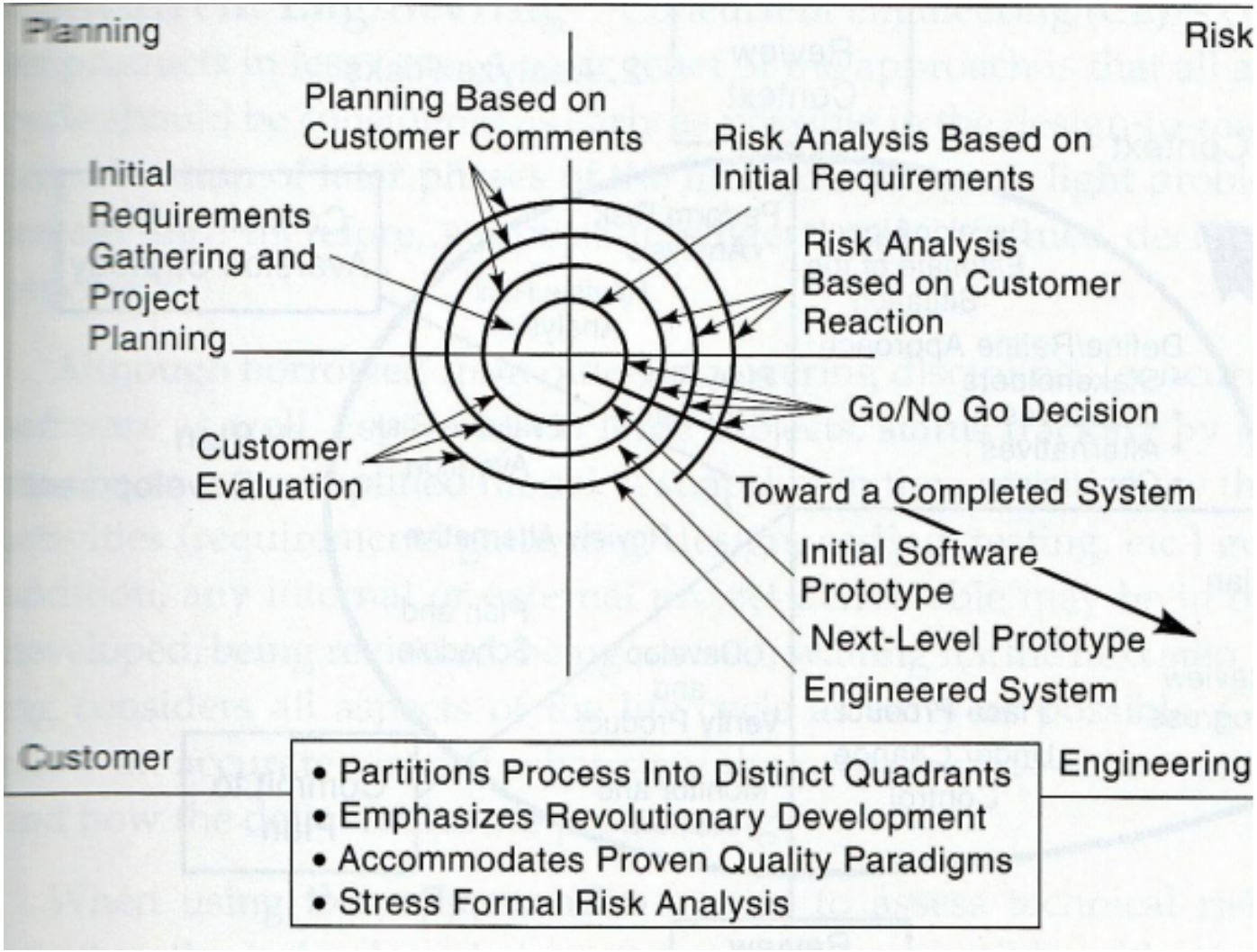
prototyping

- Splits large effort into chunks
  - Implement high risk functions first

incremental


DETERMINE OBJECTIVES, ALTERNATIVES, AND CONSTRAINTS
EVALUATE ALTERNATIVES, IDENTIFY AND RESOLVE RISKS
Risk Analysis
Support and Maintenance Objectives, Alternatives, and Constraints
Risk Analysis
Updated Operational Prototyping
Implementation Objectives, Alternatives, and Constraints
Design Objectives, Alternatives, and Constraints
Risk Analysis
Operational Prototyping
System/Product Objectives, Alternatives, and Constraints
Risk Analysis
Design Assessment Prototyping
Risk Analysis
Demonstration Prototyping
Project Definition
Conceptual Prototyping
Product Review
Design Review
Rqmts Review
System Review
Concept of Operation System Software Spec.
Simulations, and Models, Benchmarks
Engineering and Project Planning
Software Require-ments Spec, Updated System Software Specification
Detailed Design
Updated Detailed Design
Design and Development Transition Planning
Enhanced Operational Capability Integration, Activation, and Training Planning
CSCI Integration and Test Site Activation Training Planning
Software Architecture and Preliminary SDDs
Code
Unit Test
Code
Unit Test
Integration and Test
Integration and Test
IOC DELIVERY
Qualification Testing
Formal Testing
User Acceptance Test and Training
FOC DELIVERY
FCA/PCA
PLAN NEXT PHASE
DEVELOP NEXT-LEVEL PRODUCT

Simplified View of Spiral



When to use Spiral?

- Requirements are too complex or evolving
- Medium to high risk projects
- New technology or proof of concept only
- See system early on
  - How many loops?
- Ex: aerospace (Mars rover), defense apps, engineering projects

Agile Development

- Lightweight approach
  - Adapt quickly and painlessly to evolving requirements
  - Use short, frequent iterations (~15 days)
  - Highly collaborative
- Does not require PM
  - Self-organizing teams
  - Minimal management
  - Some still use a PM for communication purposes
- Well-known methods:
  - Scrum
  - Extreme programming (XP)

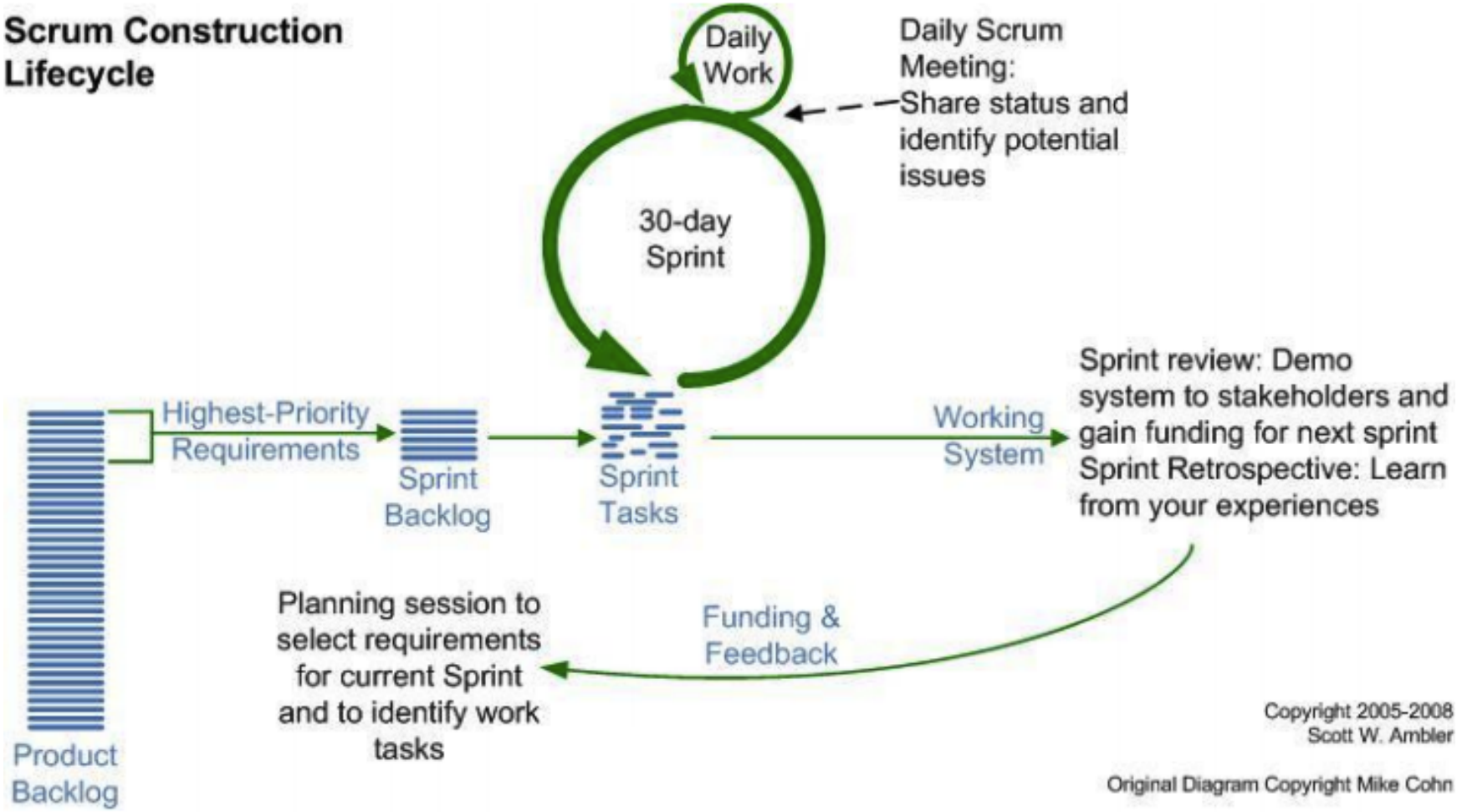
Major Disadvantages

- Examples?

Major Disadvantages

- Difficult to integrate inexperienced programmers
  - May fall back to “cowboy” coding
- Hard to estimate schedule completion
- Increases risk of scope creep
  - How many iterations?
- Can be inefficient
  - No authority to make decisions

Scrum


Scrum Construction Lifecycle
Daily Work
Daily Scrum Meeting: Share status and identify potential issues
30-day Sprint
Highest-Priority Requirements
Sprint Backlog
Sprint Tasks
Working System
Sprint review: Demo system to stakeholders and gain funding for next sprint
Sprint Retrospective: Learn from your experiences
Product Backlog
Planning session to select requirements for current Sprint and to identify work tasks
Funding & Feedback
Copyright 2005-2008 Scott W. Ambler
Original Diagram Copyright Mike Cohn

XP

- Practices common sense principles to the extreme
- Follows SE practices that support high quality software
- "Code for today, not for tomorrow"
  - Just-in-time design
- Emphasize customer involvement
- Test-driven development
  - Write test stubs first then fill in code
  - Continuous integration testing
- Pair programming environment
- Short iterations with fact delivery

How to choose an SDLC

- Requirements
  - Defined and stable early on?
  - Is early functionality a requirement?
- Project team
- User involvement
- Project type/risk

How to choose an SDLC

- Requirements
- Project team
  - Training available?
  - Likely to have changing team members?
- User involvement
- Project type/risk

How to choose an SDLC

- Requirements
- Project team
- User involvement
  - Do they want to be involved? How much?
  - Does client want to track progress?
- Project type/risk

How to choose an SDLC

- Requirements
- Project team
- User involvement
- Project type/risk
  - Funding stable through lifecycle?
  - Reusable components available?

TABLE 4–2 (Continued)
Selecting a Life Cycle Model Based on Characteristics of the Project Team

Project Team	Waterfall	V-Shaped	Prototype	Spiral	RAD	Incremental
Will the project manager closely track the team's progress?	Yes	Yes	No	Yes	No	Yes
Is ease of resource allocation important?	Yes	Yes	No	No	Yes	Yes
Does the team accept peer reviews and inspections, management/ customer reviews, and milestones?	Yes	Yes	Yes	Yes	No	Yes

TABLE 4–3
Selecting a Life Cycle Model Based on Characteristics of the User Community

User Community	Waterfall	V-Shaped	Prototype	Spiral	RAD	Incremental
Will the availability of the user representatives be restricted or limited during the life cycle?	Yes	Yes	No	Yes	No	Yes
Are the user representatives new to the system definition?	No	No	Yes	Yes	No	Yes
Are the user representatives experts in the problem domain?	No	No	Yes	No	Yes	Yes
Do the users want to be involved in all phases of the life cycle?	No	No	Yes	No	Yes	No
Does the customer want to track project progress?	No	No	Yes	Yes	No	No

Project Type and Risk	Waterfall	V-Shaped	Prototype	Spiral	RAD	Incremental
Does the project identify a new product direction for the organization?	No	No	Yes	Yes	No	Yes
Is the project a system integration project?	No	Yes	Yes	Yes	Yes	Yes
Is the project an enhancement to an existing system?	No	Yes	No	No	Yes	Yes
Is the funding for the project expected to be stable throughout the life cycle?	Yes	Yes	Yes	No	Yes	No
Is the product expected to have a long life in the organization?	Yes	Yes	No	Yes	No	Yes
Is high reliability a must?	No	Yes	No	Yes	No	Yes
Is the system expected to be modified, perhaps in ways not anticipated, post-deployment?	No	No	Yes	Yes	No	Yes

TABLE 4–4 (Continued)
Selecting a Life Cycle Model Based on Characteristics of Project Type and Risk

Project Type and Risk	Waterfall	V-Shaped	Prototype	Spiral	RAD	Incremental
Is the schedule constrained?	No	No	Yes	Yes	Yes	Yes
Are the module interfaces clean?	Yes	Yes	No	No	No	Yes
Are reusable components available?	No	No	Yes	Yes	Yes	No
Are resources (time, money, tools, people) scarce?	No	No	Yes	Yes	No	No