

# Software Engineering

COEN 174

Ron Danielson

Software Process Models

Chapter 4

# Objectives

- Understand what a software process model is
- Know why SPM are important
- Know the advantages and disadvantages of the major SPMs
  - Waterfall
  - Incremental
  - Spiral

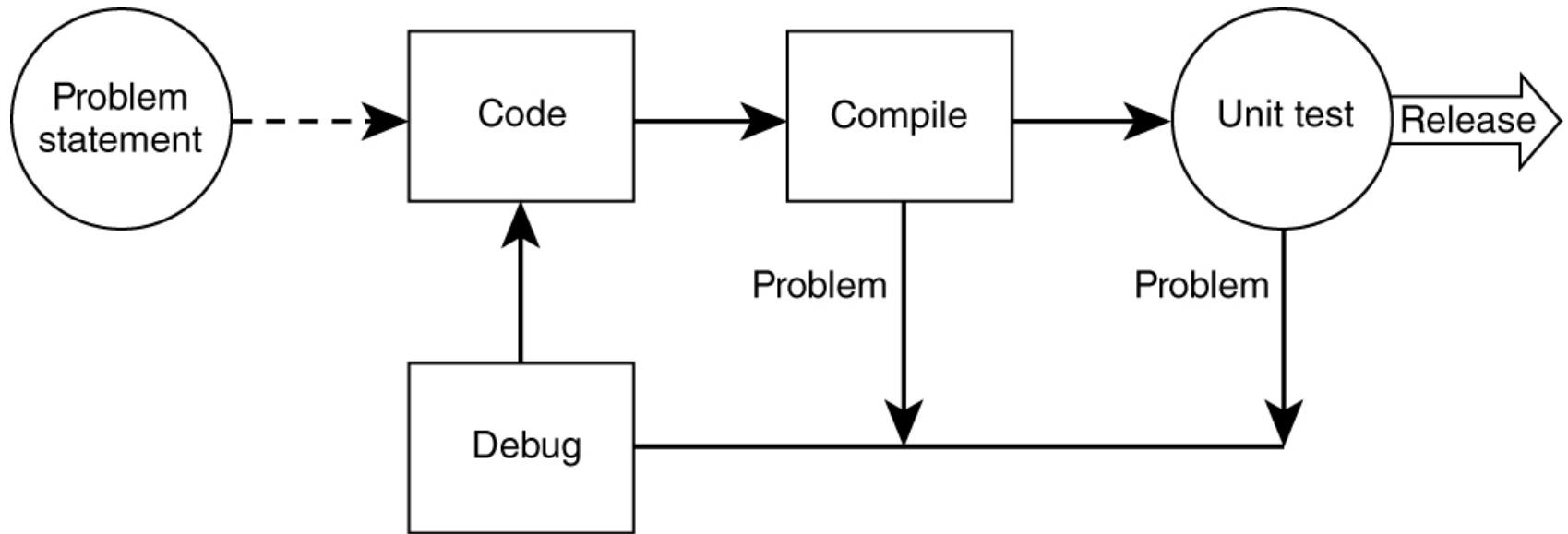
# Process Models

- A **description**
  - Tasks that need to be performed
  - Preconditions and postconditions for each task
  - Inputs and outputs for each task
  - Sequence of performance
  - Conditions under which performance takes place
  - Who does what
- To achieve “desired results”
- Or, a **structure for the development of products**

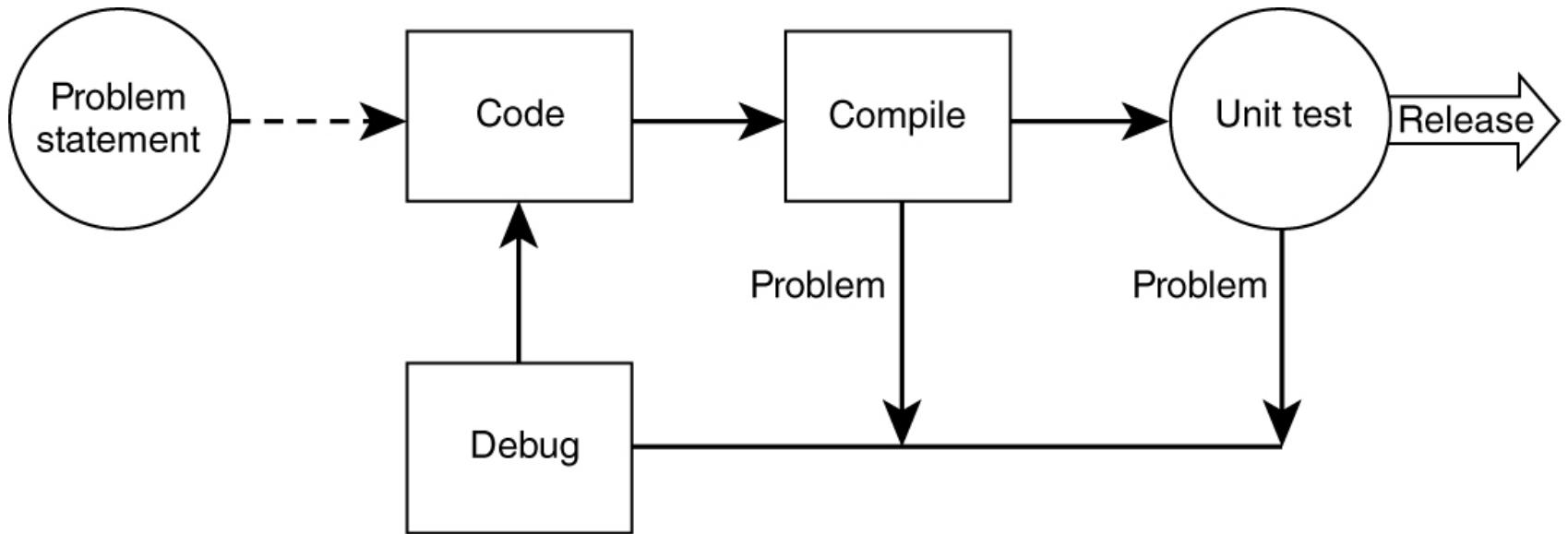
# Why Have a Process Model?

- To help with **coordinating** and **controlling**
  - **Tasks** to be performed
  - **Personnel** who perform the tasks
- Goals include to make the process
  - Consistent
  - Repeatable
  - Transparent
  - Manageable
- Is a process needed for ALL software development?
- **No** SW process fits **all** SW projects

# A Simple SW Process Model



# A Simple SW Process Model



- This process ignores
  - Requirements
  - Design
  - Formal testing
  - Documentation
  - Packaging

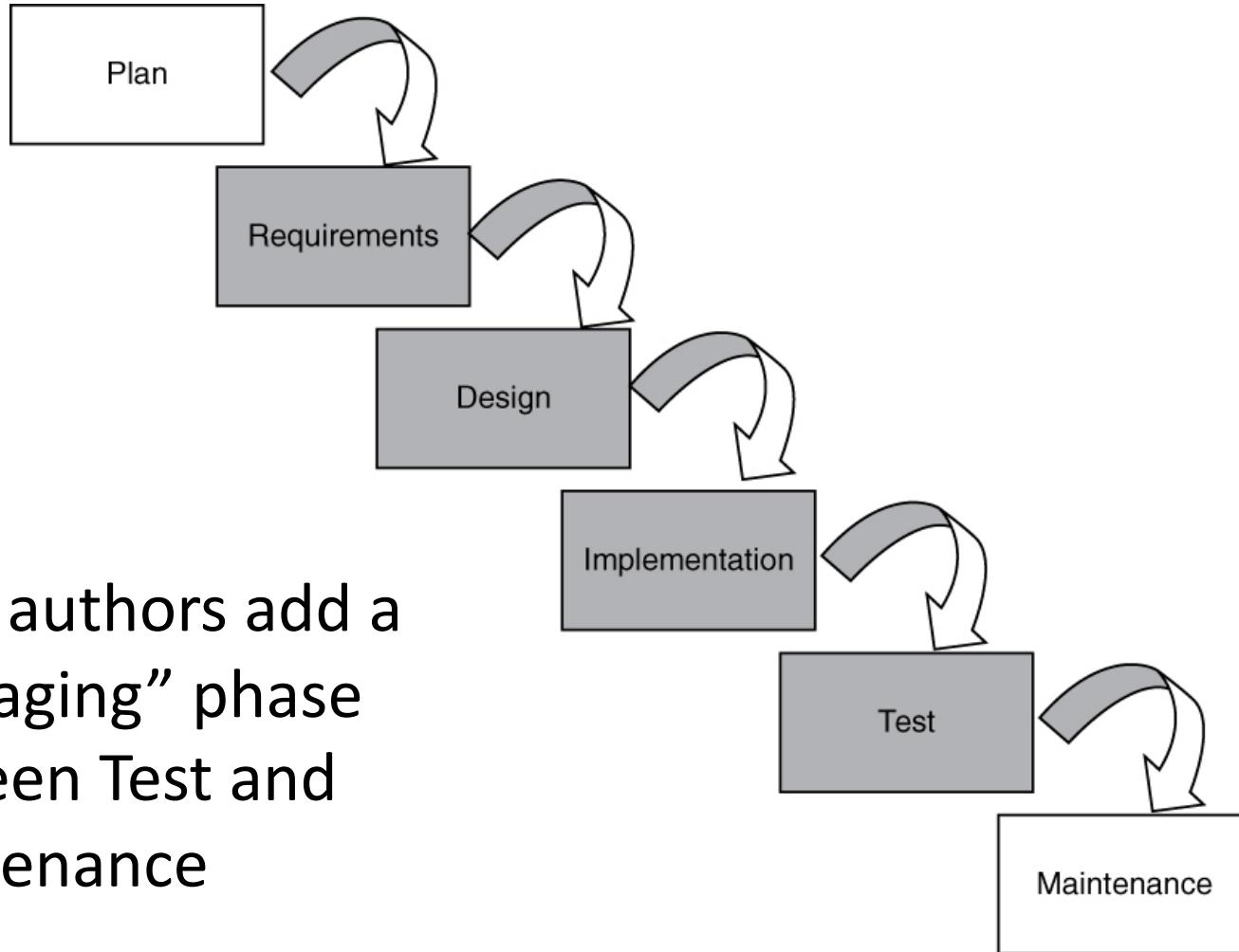
# Extending the Simple Process

- Larger and more complex projects must
  - Clarify and stabilize the requirements
  - Test more functionalities
  - Design more carefully
  - Reuse existing software and deploy tools
    - Database, network, code control
  - Coordinate more people
- Need a process to minimize the chance of failure
- Like all our “models” these are idealizations of what really happens

# A Waterfall



# The Waterfall SW Process



# The Waterfall SW Process (cont.)

- Essentially, the steps of the traditional engineering process model
- One of the first SW process models
- Advantages
  - Simplicity
  - Familiarity
  - Visibility
  - Easy to monitor where you are on the path to completion because process is rigid
  - Facilitates allocation of resources

# The Waterfall SW Process (cont.)

- Disadvantages
  - Assumes all requirements are known in advance
  - Doesn't incorporate changes well
  - Unidirectional (which isn't realistic)
  - Problems are found late in process
  - Lack of parallelism
  - Some resources idle for part of process
  - No feedback to client; client needs LOTS of patience
- Good for projects that are small and have very well-defined requirements

# V Model

Requirements engineering

System design

Program design

Implementation

Acceptance testing

System testing

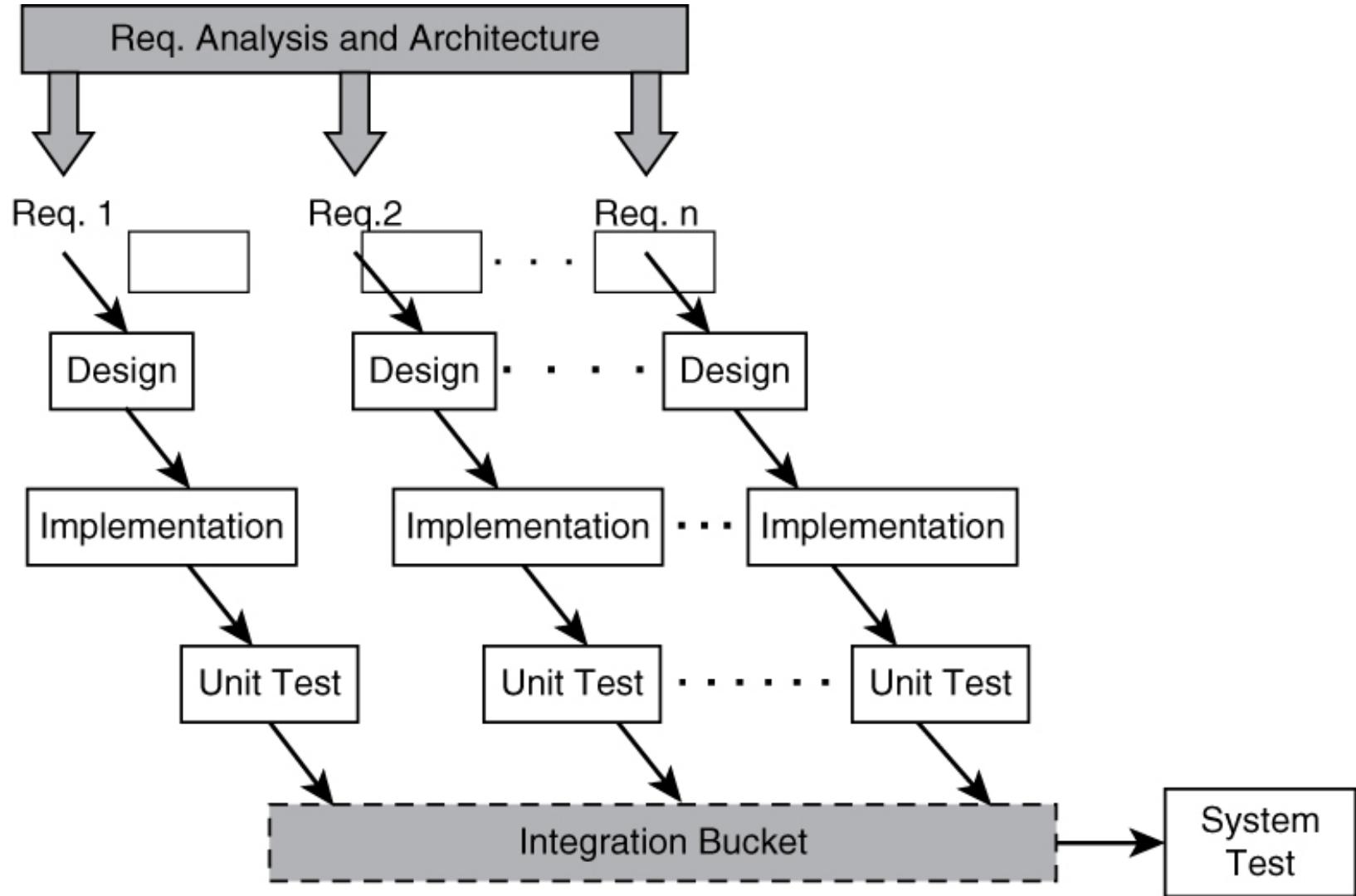
Unit testing

Variation of waterfall, with emphasis on the purpose of testing

# Incremental Models

- Implementing all the functionality of a large SW project can be daunting
- Modification is to divide whole system into a set of smaller, independent components
  - Implement independently
  - As completed, merge into single integration of whole system
- Advantages
  - Reduced complexity of components
  - Risk containment – trouble with one component wouldn't delay the others

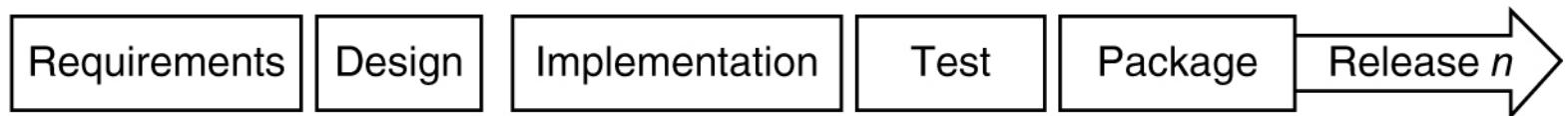
# Continuous Integration



# Continuous Integration (cont.)

- Advantages
  - Reduced complexity of components
  - Parallel development should be faster
  - Easier management of smaller streams
  - Risk containment – trouble with one component wouldn't delay the others
- Disadvantages
  - Poorer visibility into whole process
  - If linkages are needed between components can increase complexity, not decrease
  - Harder to create a unified design

# Multiple Releases



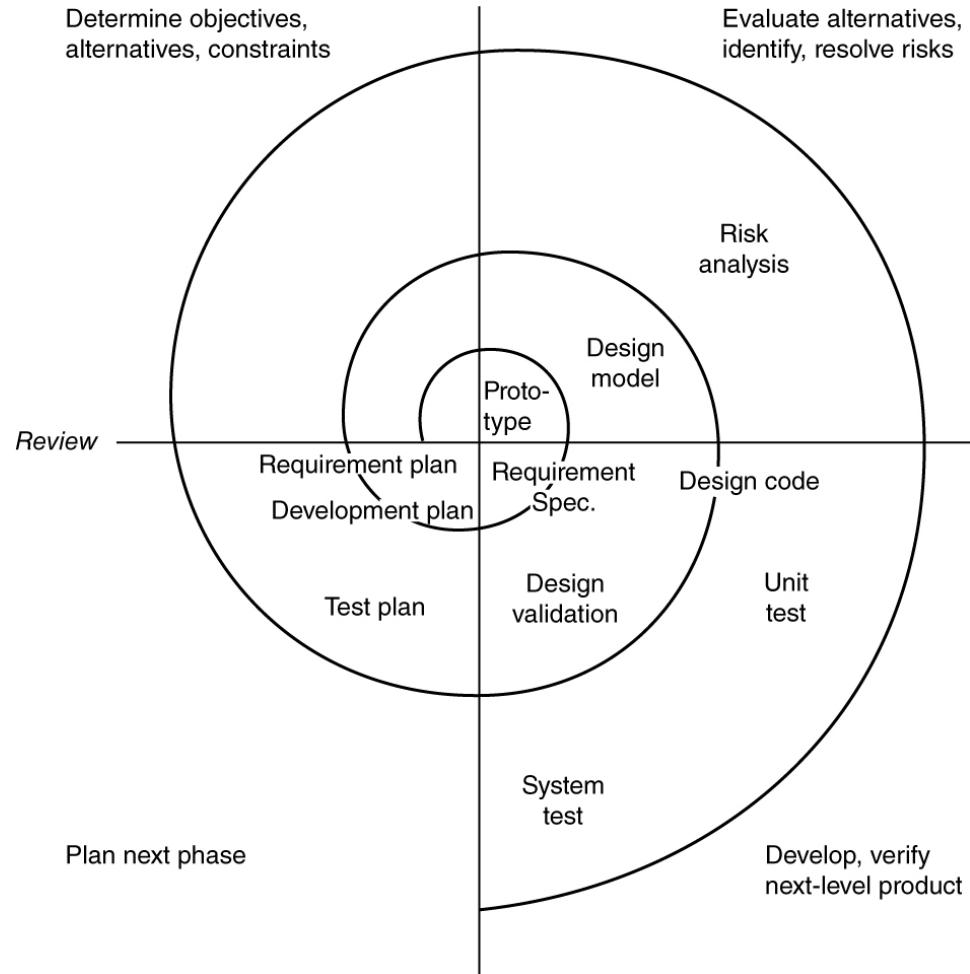
# Multiple Releases (cont.)

- First release contains core (or critical) functionality
- Subsequent releases add functionality
- Advantages
  - Gets product to market faster
  - Implementation by client can be easier since impact is compartmentalized
  - Bug fixes integrated with next release
- Disadvantages
  - Dependency between components problematic
  - Design of subsequent release may impact earlier releases

# Incremental Overall

- Adapts better to changes in requirements
- Potentially uses fewer resources
- Requires that the overall problem can be decomposed into pieces
  - Pieces need to be separable
  - If not, complexity increases
- Managing overlapping releases can be more complicated
- Less visibility into whole process

# Spiral



# Spiral (cont.)

- Attempt to reduce risks associated with SW development
  - Waterfall is a “big bang” process with heavy up-front investment
  - In spiral, product gradually becomes more mature as wind through successive spirals
    - Risk analyzed frequently, mitigated
    - Or, project abandoned
  - Heavy use of prototyping and modeling
  - Requires more client involvement/communication than prior models

# Spiral (cont.)

- Identify objectives, alternatives, constraints for this iteration
- Evaluate alternatives relative to objectives and constraints, identify and resolve risks
  - Cost-benefit analysis
    - Cost vs. value
    - Time to develop
    - Resources required
    - Impact on existing system
- Development
- Validate achievement of objective and plan for next iteration (client involvement)

# Spiral (cont.)

- Phases can mean different things on different iterations through the spiral
  - Development could mean build a prototype, validate a design, write code, ...
  - Planning could be for requirements solicitation, development, testing, ...
- OK to return to and rework previous decisions
- Releases don't overlap
- Usually gives client something to react to early
- Risk analysis can be tricky

# Spiral (cont.)

- Advantages
  - Risks managed early and throughout the process
    - Prototyping and modeling an integral part
  - Software evolves as the project progresses
    - Iterative and evolutionary process
  - Planning built into process
- Disadvantages
  - Complicated to use
  - May be overkill for small projects
  - Most SW developers not well-versed in risk analysis

# Rational Unified Process

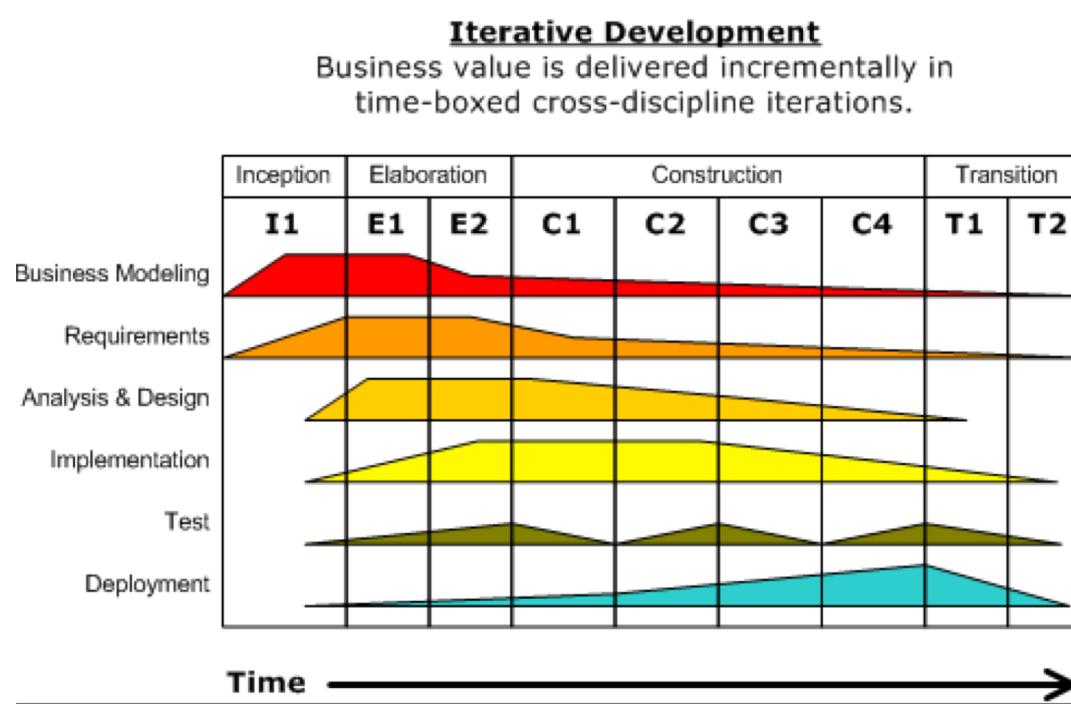
- More recent SW development process
  - Distills essential elements of some earlier processes
  - Also based on UML
- Six practices or principles
  - Develop iteratively, with risk as primary driver
  - Manage requirements
  - Employ a component-based architecture
  - Model SW visually
  - Verify quality continuously
  - Control changes
- Architecture-centric
  - Architecture in this case is the high-level design elements that span use cases
    - E.g., user interface standards, error handling standards, ...

# Rational Unified Process (cont.)

- Driven by use case analysis and requirements
  - Each major use case represents a key component of the system
  - Architecture refined to accommodate major use cases
  - Elaboration of use cases can also drive changes in the architecture
- Both iterative and incremental
  - Develop large software in smaller increments
  - Develop each increment in iterative fashion
    - All use cases and requirements
    - Deal with major risks in the next increment
    - Rinse and repeat

# Rational Unified Process (cont.)

- Four phases
  - Inception
  - Elaboration
  - Construction
  - transition



# Rational Unified Process (cont.)

- Inception
  - Establish feasibility
  - Make business case
  - Establish scope and goals
  - Establish critical use cases and major scenarios
  - Establish some architecture and design alternatives
  - Estimate schedule and required resources
  - Plan the implementation, testing integration and configuration methodologies
  - Estimate risks
  - Build one or more prototypes

# Rational Unified Process (cont.)

- Elaboration
  - Specify requirements in greater detail
  - Create architectural baseline
  - Perform iterative implementation of core architecture
  - Establish implementation, test and integration planes
  - Establish test scenarios
  - Refine risk assessment and resolve highest risk items
  - Define metrics
  - Refine project plan, including detailed plan for construction

# Rational Unified Process (cont.)

- Construction
  - Complete remaining requirements
  - Complete iterative implementation of remaining requirements
  - Test and prepare system for deployment
  - Alpha release
  - Establish activities remaining to complete project

# Rational Unified Process (cont.)

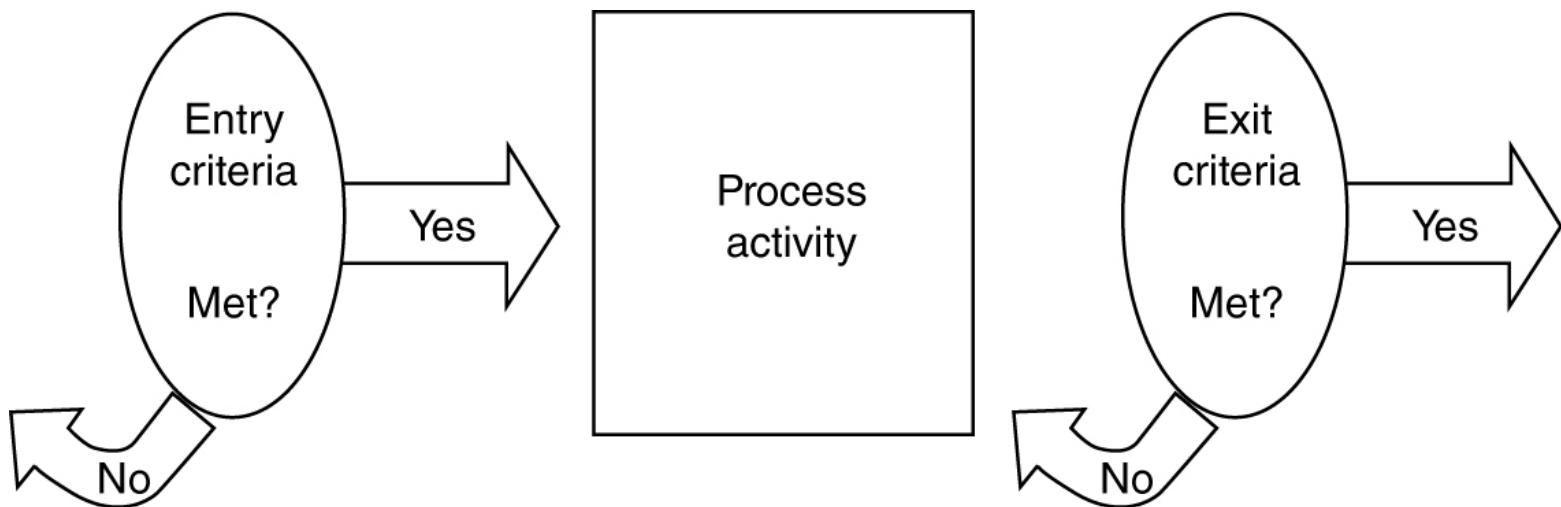
- Transition
  - Establish final SW product
  - Correct defects
  - Conduct beta tests
  - Establish support readiness
    - Create user manuals
    - Training materials for end users and customer support
  - Gain agreement on release and deployment
  - Conduct lessons learned

# Rational Unified Process (cont.)

- Disadvantages
  - Complicated process
  - Requires support and management attention
  - Heavy process may be overkill for many projects
- Advantages
  - Good for BIG projects
  - Can deal with complex requirements
  - Accounts for most aspects of a project
  - Mature process widely used

# Entry and Exit Criteria

- We've talked about various activities in a number of SW process models
  - To make each step meaningful, there need to be defined entry criteria and exit criteria for each



# Entry and Exit Criteria (cont.)

- Entry criteria
  - List and describe required
    - Artifacts
    - People
    - Tools
    - Definition of activity to be performed
- Exit criteria
  - List and describe required
    - Artifacts
    - conditions

# Choosing Model and Structure

You are a program manager who has the task of salvaging a project that is vital to the success of your company. Upper-level management has given you an unlimited budget and has suggested that you assemble your new team by pulling people from other projects. However, a competitor is also actively working on developing a similar product and has already hired several former employees of your firm.

# Choosing Model and Structure

You are a program manager for an aerospace firm that develops software for missile systems. Your firm has recently been awarded the contract for developing guidance systems for the latest generation of cruise missiles. The military has strict guidelines that must be met for these systems, along with a firm deadline.