

Software Life Cycle

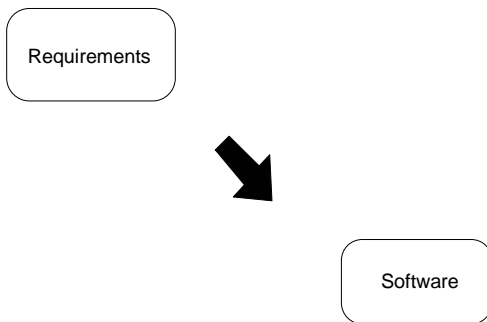
Sources:

1. B. Bruegge and A. H. Dutoit, Object-Oriented Software Engineering: Using UML, Patterns, and Java (Chapter 15)
2. Bob Hughes et al., Software Project Management 4th edition (Chapter 4)

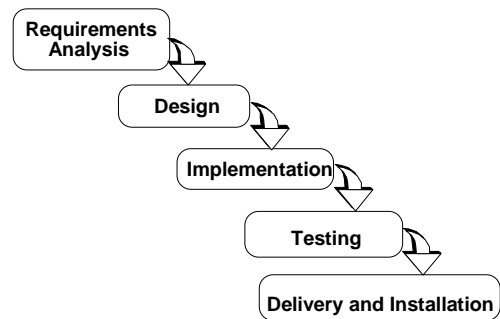
Outline

- Software Life Cycle
 - Waterfall model and its problems
 - Pure Waterfall Model
 - V-Model
 - Iterative process models
 - Boehm's Spiral Model
 - Entity-based models
 - Issue-based Development Model (Concurrent Development)

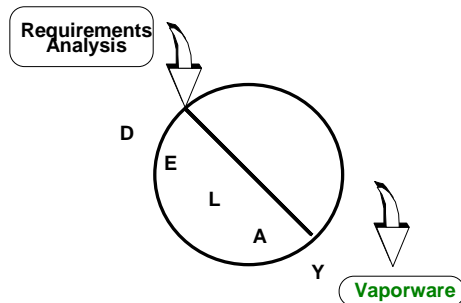
What we intend



How it should go: Our plan of attack



How it often goes



Inherent Problems with Software Development

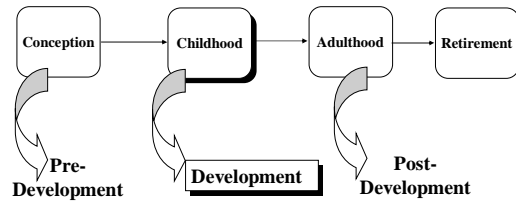
- Requirements are complex
 - The client does not know the functional requirements in advance
- Requirements may be changing
 - Technology enablers introduce new possibilities to deal with nonfunctional requirements
- Frequent changes are difficult to manage
 - Identifying milestones and cost estimation is difficult
- There is more than one software system
 - New system must be backward compatible with existing system ("legacy system")
 - Phased development: Need to distinguish between the system under development and already released systems
- Let's view these problems as the nonfunctional requirements for a system that supports software development!
 - This leads us to software life cycle modeling

Definitions

- Software lifecycle modeling: Attempt to deal with complexity and change
- Software lifecycle:
 - Set of activities and their relationships to each other to support the development of a software system
- Software development methodology:
 - A collection of techniques for building models - applied across the software lifecycle

Software Life Cycle

- Software construction goes through a progression of states



Typical Software Lifecycle Questions

- Which activities should I select for the software project?
- What are the dependencies between activities?
 - Does system design depend on analysis? Does analysis depend on design?
- How should I schedule the activities?
 - Should analysis precede design?
 - Can analysis and design be done in parallel?
 - Should they be done iteratively?

Identifying Software Development Activities

- For finding activities and dependencies we can use the same modeling techniques when modeling a system such as creating scenarios, use case models, object identification, drawing class diagrams, activity diagrams
- Questions to ask:
 - What is the problem?
 - What is the solution?
 - What are the mechanisms that best implement the solution?
 - How is the solution constructed?
 - Is the problem solved?
 - Can the customer use the solution?
 - How do we deal with changes that occur during the development? Are enhancements needed?

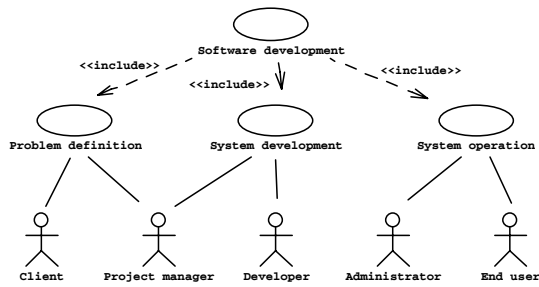
Possible Identification of Software Development Activities

Requirements Analysis	What is the problem?	Problem Domain
System Design	What is the solution?	
Program Design	What are the mechanisms that best implement the solution?	Implementation Domain
Program Implementation	How is the solution constructed?	
Testing	Is the problem solved?	
Delivery	Can the customer use the solution?	
Maintenance	Are enhancements needed?	

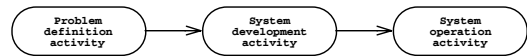
Alternative Identification of Software Development Activities

Requirements Analysis	What is the problem?	Problem Domain
System Design	What is the solution?	
Object Design	What is the solution in the context of an existing hardware system?	Implementation Domain
Implementation	How is the solution constructed?	

Software Development as Application Domain: A Use Case Model

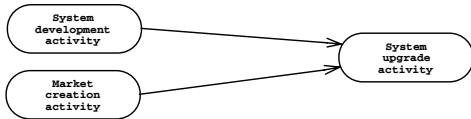


Activity diagram for the same life cycle model



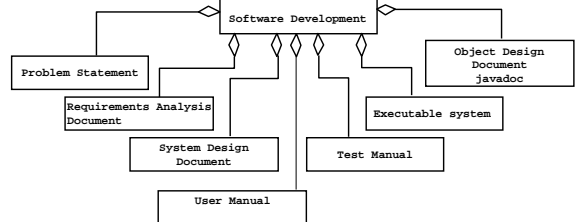
Software development goes through a linear progression of states called software development activities

Another simple life cycle model

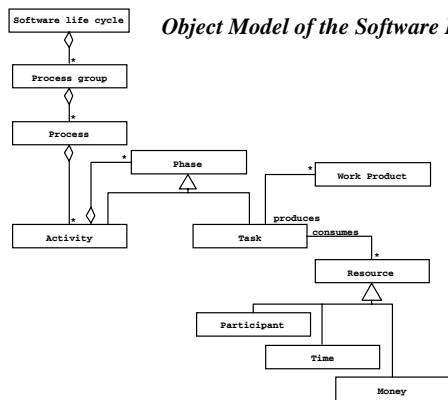


System Development and Market creation can be done in parallel and Must be done before the system upgrade activity

Software Development as Application Domain: Simple Object Model



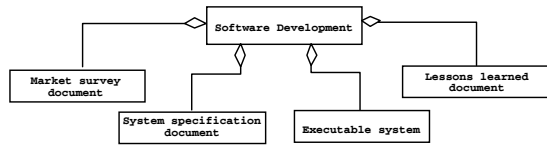
Object Model of the Software Life Cycle



Two Major Views of the Software life cycle

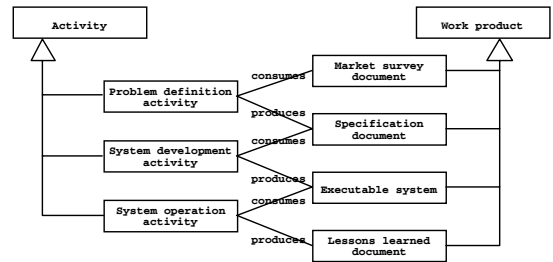
- ❖ Activity-oriented view of a software life cycle
 - all the examples so far
- ❖ Entity-oriented view of a software life cycle

Entity-centered view of software development

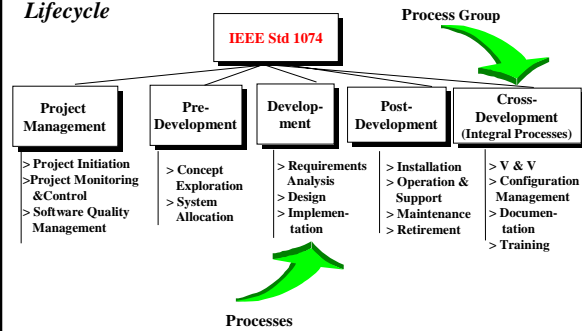


Software development consists of the creation of a set of deliverables

Combining activities and entities in one view

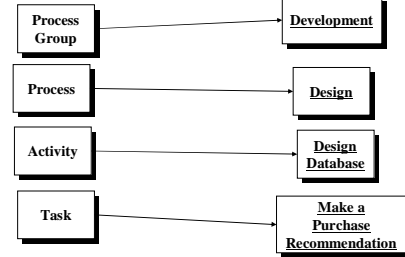


IEEE Std 1074: Standard for Software Lifecycle



Processes, Activities and Tasks

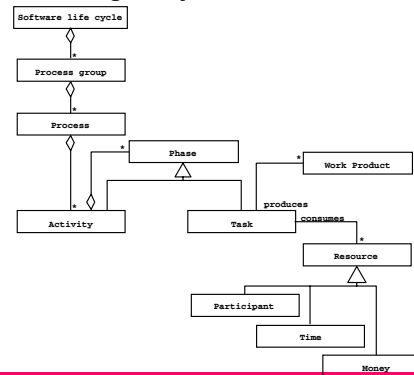
- ◆ Process Group: Consists of Set of Processes
- ◆ Process: Consists of Activities
- ◆ Activity: Consists of sub activities and tasks



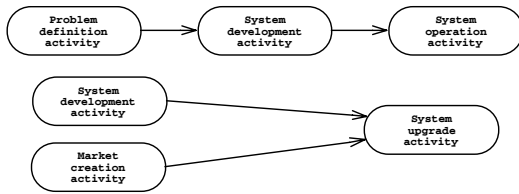
Example

- ◆ The Design Process is part of Development
- ◆ The Design Process consists of the following Activities
 - ◆ Perform Architectural Design
 - ◆ Design Database (If Applicable)
 - ◆ Design Interfaces
 - ◆ Select or Develop Algorithms (If Applicable)
 - ◆ Perform Detailed Design (= Object Design)
- ◆ The Design Database Activity has the following Tasks
 - ◆ Review Relational Databases
 - ◆ Review Object-Oriented Databases
 - ◆ Make a Purchase recommendation
 - ◆

UML Class Diagram of the IEEE Standard



Modeling Dependencies in a Software Lifecycle

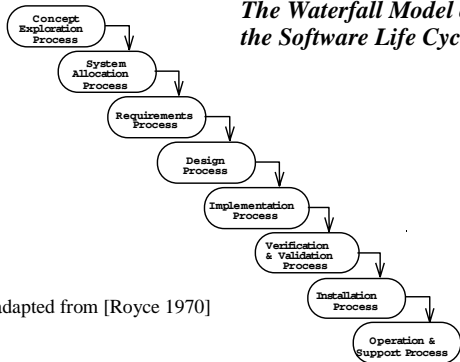


- Note that the dependency association can mean one of two things:
 - Activity B depends on Activity A
 - Activity A must temporarily precede Activity B
- Which one is right?

Life-Cycle Model: Variations on a Theme

- Many models have been proposed to deal with the problems of defining activities and associating them with each other
- The waterfall model
 - First described by Royce in 1970
- There seem to be at least as many versions as there are authorities - perhaps more

The Waterfall Model of the Software Life Cycle

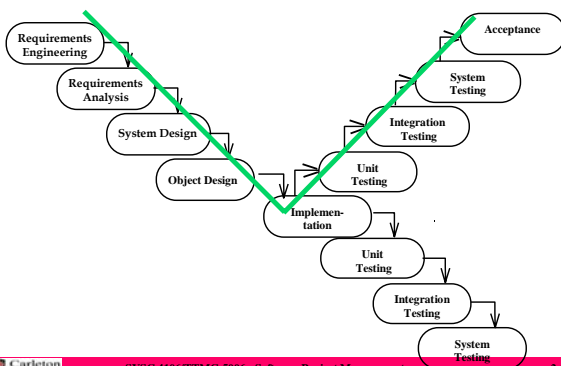


adapted from [Royce 1970]

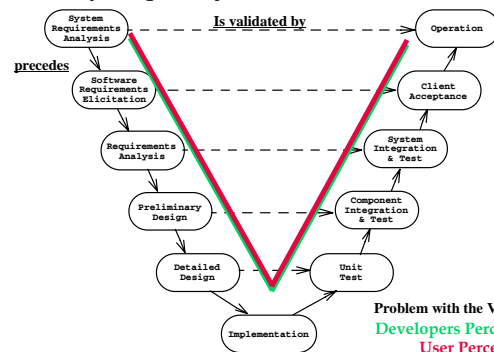
Problems with Waterfall Model

- Managers love waterfall models:
 - Nice milestones
 - No need to look back (linear system), one activity at a time
 - Easy to check progress : 90% coded, 20% tested
- Different stakeholders need different abstractions
 - => V-Model
- Software development is iterative
 - During design problems with requirements are identified
 - During coding, design and requirement problems are found
 - During testing, coding, design & requirement errors are found
 - => Spiral Model
- System development is a nonlinear activity
 - => Issue-Based Model

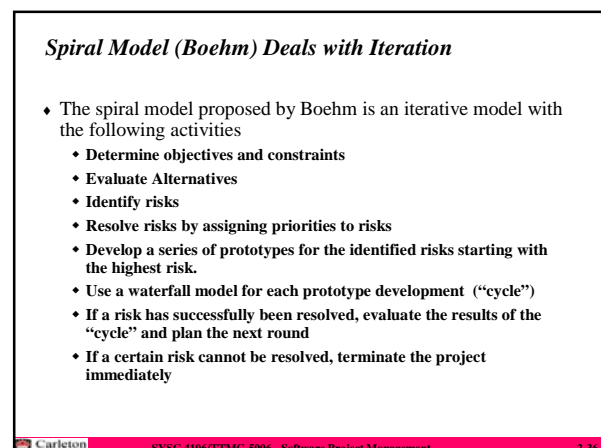
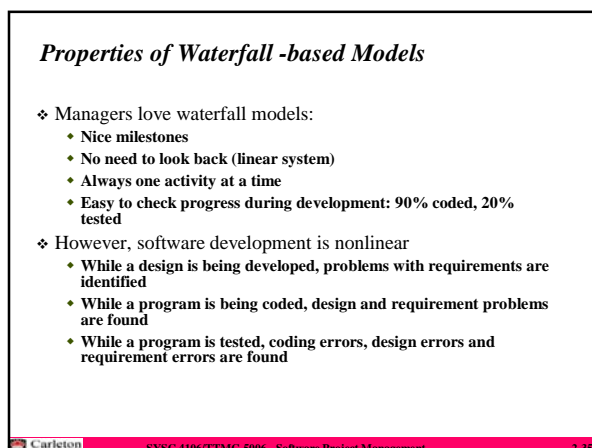
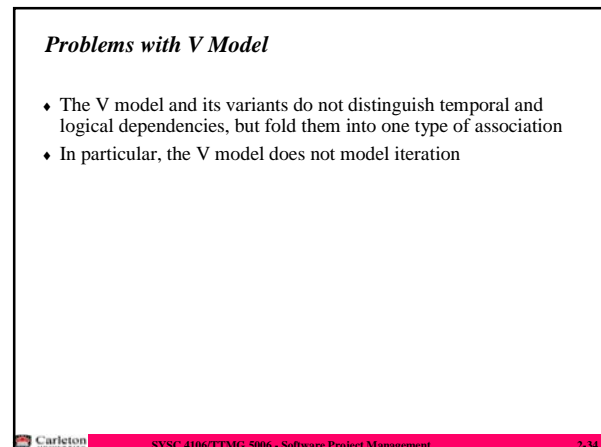
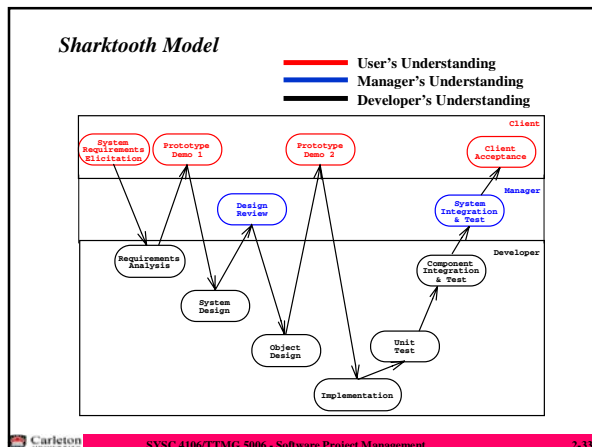
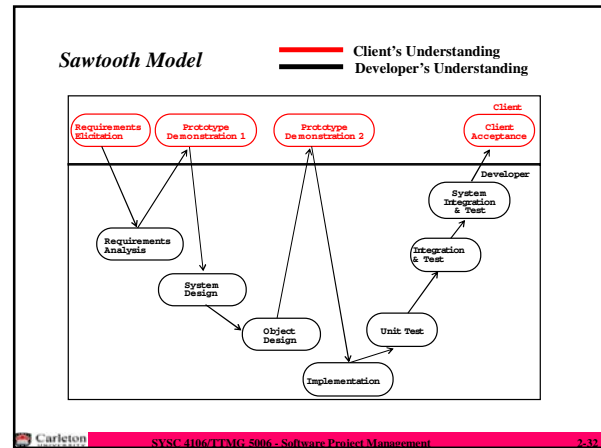
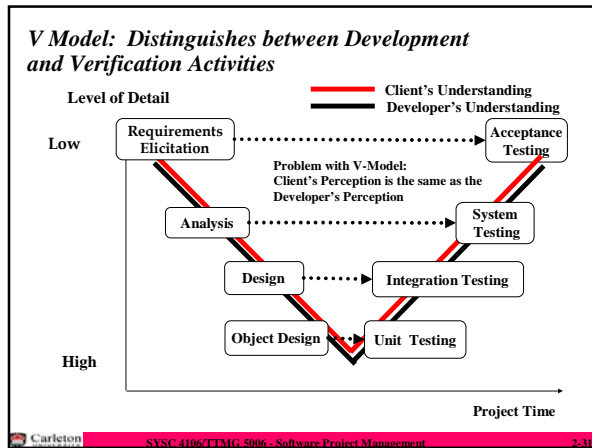
From the Waterfall Model to the V Model



Activity Diagram of a V Model



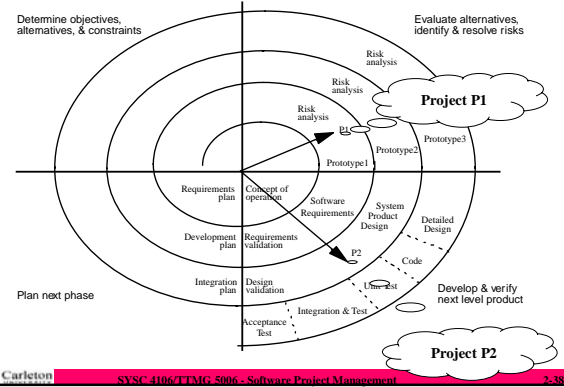
Problem with the V-Model:
Developers Perception =
User Perception



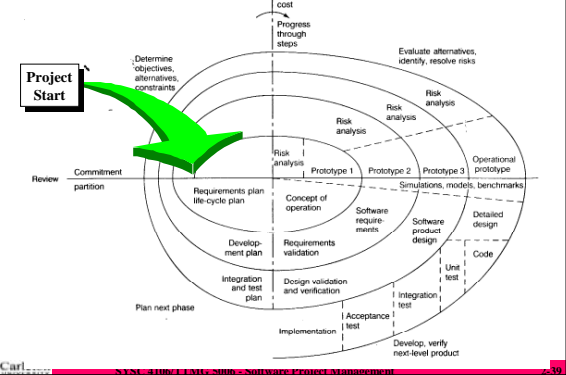
Activities (Cycles) in Boehm's Spiral Model

- ❖ Concept of Operations
 - ❖ Software Requirements
 - ❖ Software Product Design
 - ❖ Detailed Design
 - ❖ Code
 - ❖ Unit Test
 - ❖ Integration and Test
 - ❖ Acceptance Test
 - ❖ Implementation
- ❖ For each cycle go through these activities
 - Quadrant IV: Define objectives, alternatives, constraints
 - Quadrant I: Evaluate alternative, identify and resolve risks
 - Quadrant II: Develop, verify prototype
 - Quadrant III: Plan next "cycle"
 - ❖ The first 3 cycles are shown in a polar coordinate system.
 - The polar coordinates $r = (l, a)$ of a point indicate the resource spent in the project and the type of activity

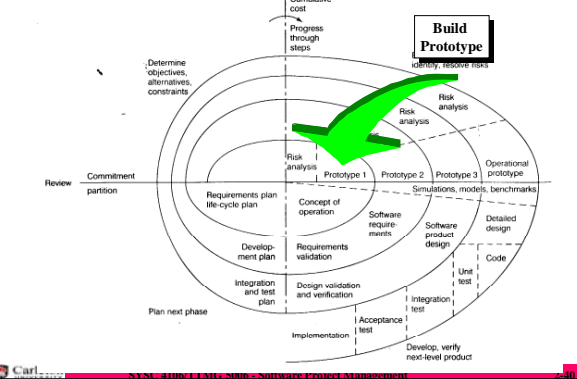
Spiral Model



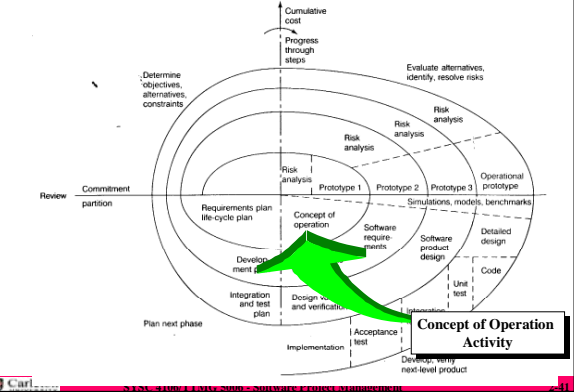
Cycle 1, Quadrant IV: Determine Objectives, Alternatives and Constraints



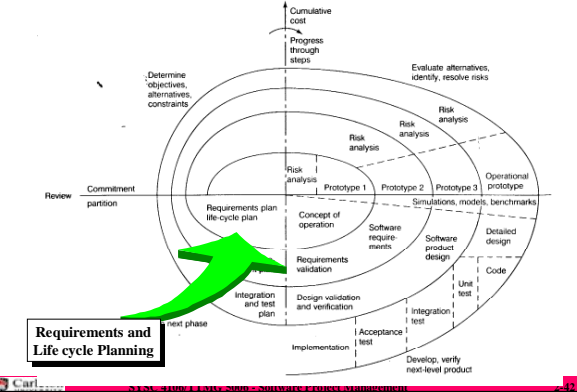
Cycle 1, Quadrant I: Evaluate Alternatives, Identify, resolve risks



Cycle 1, Quadrant II: Develop & Verify Product



Cycle 1, Quadrant III: Prepare for Next Activity



Limitations of Waterfall and Spiral Models

- ❖ Neither of these model deals well with frequent change
 - The Waterfall model assume that once you are done with a phase, all issues covered in that phase are closed and cannot be reopened
 - The Spiral model can deal with change between phases, but once inside a phase, no change is allowed
- ❖ What do you do if change is happening more frequently?
 - "The only constant is the change"

Unified Software Development Process

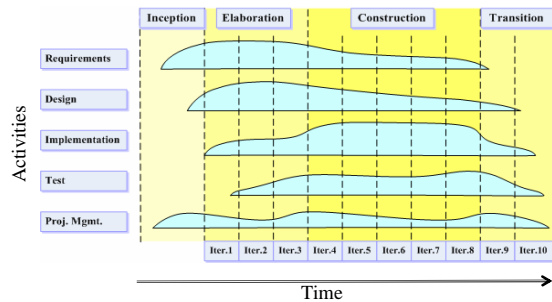
- ❖ The Unified Software Development Process (also called Unified Process) is a life cycle model proposed by Booch, Jacobson, and Rumbaugh [Jacobson et al., 1999].
- ❖ It is similar to Boehm's spiral model.
- ❖ A project consists of several cycles, each of which ends with the delivery of a product to the customer.
- ❖ Each cycle consists four phases: Inception, Elaboration, Construction, and Transition.
- ❖ Each phase itself consists of a number of iterations. Each iteration addresses a set of related use cases or mitigates some of the risks identified at the beginning of the iteration.
- ❖ The Unified Process emphasizes the staging of resources, an aspect of software development that is not captured in other life cycle models

Unified Process

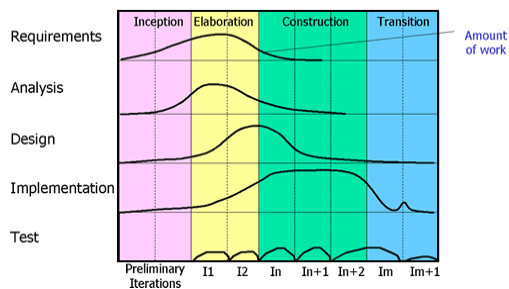
Iterative and Incremental, Use-case driven, Architecture centric Phases:

- **Inception:** The core idea is developed into a product vision. We review and confirm our understanding of the core business drivers. We want to understand the business case for why the project should be attempted. Product feasibility and project scope.
- **Elaboration:** The majority of the Use Cases are specified in detail and the system architecture is designed. "Do-Ability" of the project. We identify significant risks and prepare a schedule, staff and cost profile for the entire project.
- **Construction:** Produces a system complete enough to transition to the user. The design is refined into code.
- **Transition:** The goal is to ensure that the requirements have been met to the satisfaction of the stakeholders. Other activities include site preparation, manual completion, and defect identification and correction. The transition phase ends with a postmortem devoted to learning and recording lessons for future cycles.

Unified Process (cont.)



A different view



References

- ❖ Readings used for this lecture
 - [Humphrey 1989] Watts Humphrey, *Managing the Software Process*, SEI Series in Software Engineering, Addison Wesley, ISBN 0-201-18095-2
- ❖ Additional References
 - [Royce 1970] Winston Royce, *Managing the Development of Large Software Systems*, Proceedings of the IEEE WESCON, August 1970, pp. 1-9
 - SEI Maturity Questionnaire, Appendix E.3 in [Royce 1998], Walker Royce, *Software Project Management*, Addison-Wesley, ISBN0-201-30958-0

Summary

- ❖ Software life cycle
 - ♦ **The development process is broken into individual pieces called software development activities**
- ❖ No good model for modeling the process (black art)
 - ♦ **Existing models are an inexact representation of reality**
 - ♦ **Nothing really convincing is available today**
- ❖ Software development standards
 - ♦ **IEEE 1074**
 - ♦ **Standards help, but must be taken with a grain of salt**
 - ♦ **The standard allows the lifecycle to be tailored**