
Iterative Project Management with the Unified Process

5 Iteration Planning

Course Outline

0. Course Introduction

1. Project Initiation

2. Risk Management

3. Estimation

4. Phase Planning

5. Iteration Planning

6. PM's View of Disciplines

**7. Scope Management and
Change Control**

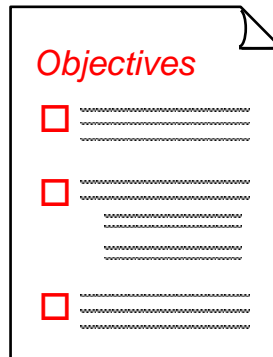
**8. Monitoring Progress:
Metrics and Reviews**

9. Communication in a Project

10. Best Practices for PM

11. Review and Wrap-Up

Lesson Objectives



- Define iterations for a project
- Estimate the size and scope of iterations
- Develop iteration evaluation criteria
- Choose appropriate iteration strategies

Iterations

- **An iteration**
 - Is a distinct sequence of activities with a plan & evaluation criteria
 - Contains all development disciplines to some extent
 - Results in an internal or external release

- **Iterations are incremental**
 - The system grows from iteration to iteration until system is complete
 - Artifacts are refined with each successive iteration

Phases

Inception		Elaboration		Construction			Transition	
Inception Iteration	Elaboration Iteration 1	Elaboration Iteration 2	Construction Iteration 1	Construction Iteration 2	Construction Iteration 3	Transition Iteration 1	Transition Iteration 2	

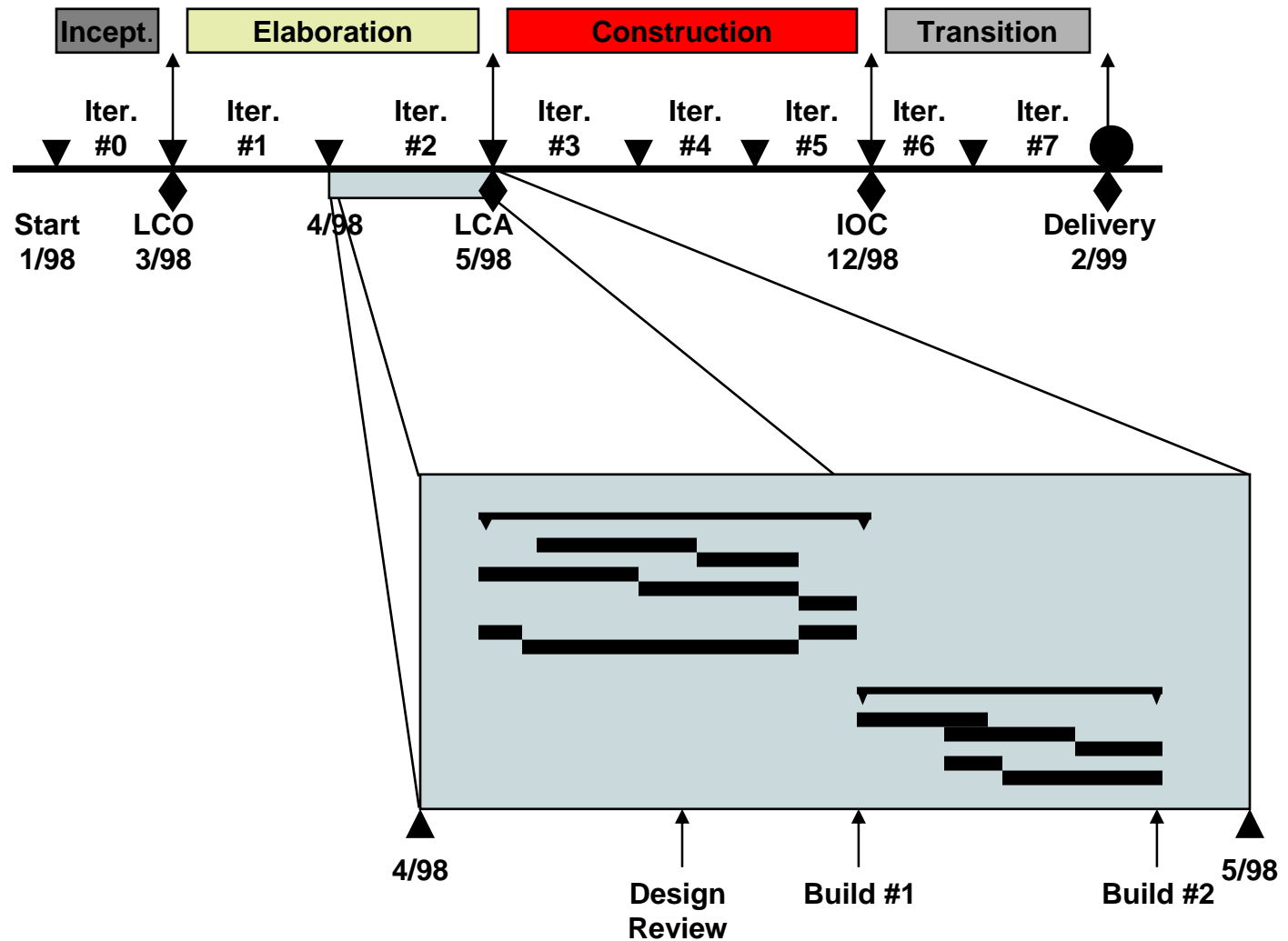
Iterations

Iteration Plan Flashes Out Phase Plan Details

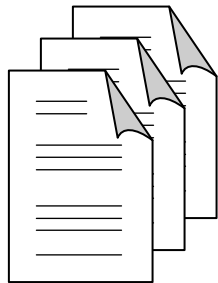
Coarse-Grained: Phase Plan



Software
Development
Plan



Fine-Grained: Iteration Plan(s)



Iteration Plan(s)

Iteration Plan

- **Created during**
 - Inception, Elaboration, & Construction phases for the next iteration
 - Transition phase if more than one iteration in Transition phase

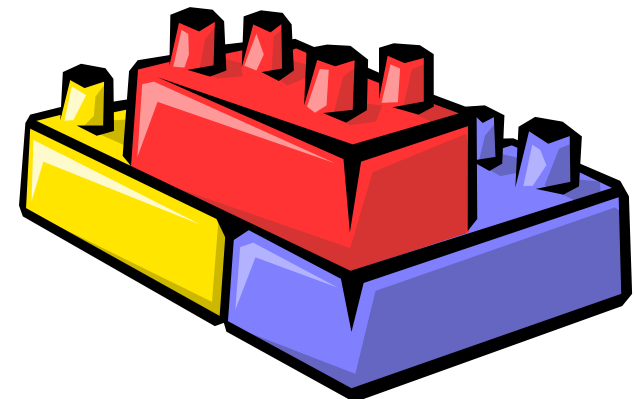
- **Consists of**
 - Goal statement
 - Use cases to be addressed
 - Risks to be mitigated
 - Architectural goals to be achieved
 - Schedule and resource assignments
 - Evaluation criteria

- List of work products to be produced appears in Phase Plan

Building an Iteration Plan

To build an iteration plan:

1. Determine Iteration Scope
2. Define Iteration Evaluation Criteria
3. Define Iteration Activities
4. Assign Responsibilities
5. Build Schedule



Determining Iteration Scope

- **Describe the level of work to be done in this Iteration. Include**
 - Use Cases
 - Non-functional requirements
 - Risks
 - Architectural Goals

- **Address the primary considerations for the current Phase:**
 - Elaboration Phase → Risk mitigation, Architectural coverage
 - Construction Phase → Business priority, Coverage, Completeness
 - Transition Phase → Performance Improvement, Bug Fixes

Prioritizing Use Cases

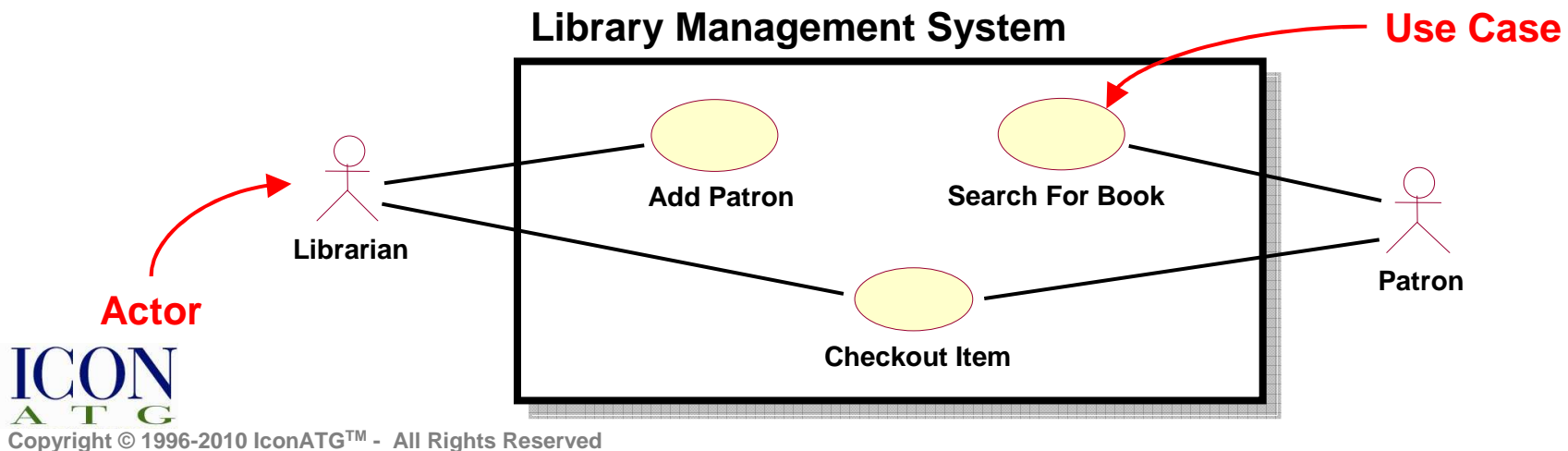
- Use cases partition system scope into manageable units that can be prioritized and assigned to iterations
- Prioritize use cases during the Inception phase
- Reassess use case priorities during Elaboration & Construction phases
- Store priorities in requirements management or other tool (Examples: RequisitePro and DOORS)

What is a Use Case?

- **A use case describes a use of the system**
 - Documents system functional requirements
 - The set of all use cases for a system are referred to as the system use case model
- **There are several artifacts in the use case model**
 - Use case diagram
 - Use case specifications
 - Actor catalog
 - Glossary
- **There is a use case specification for each use case represented on the use case diagram**

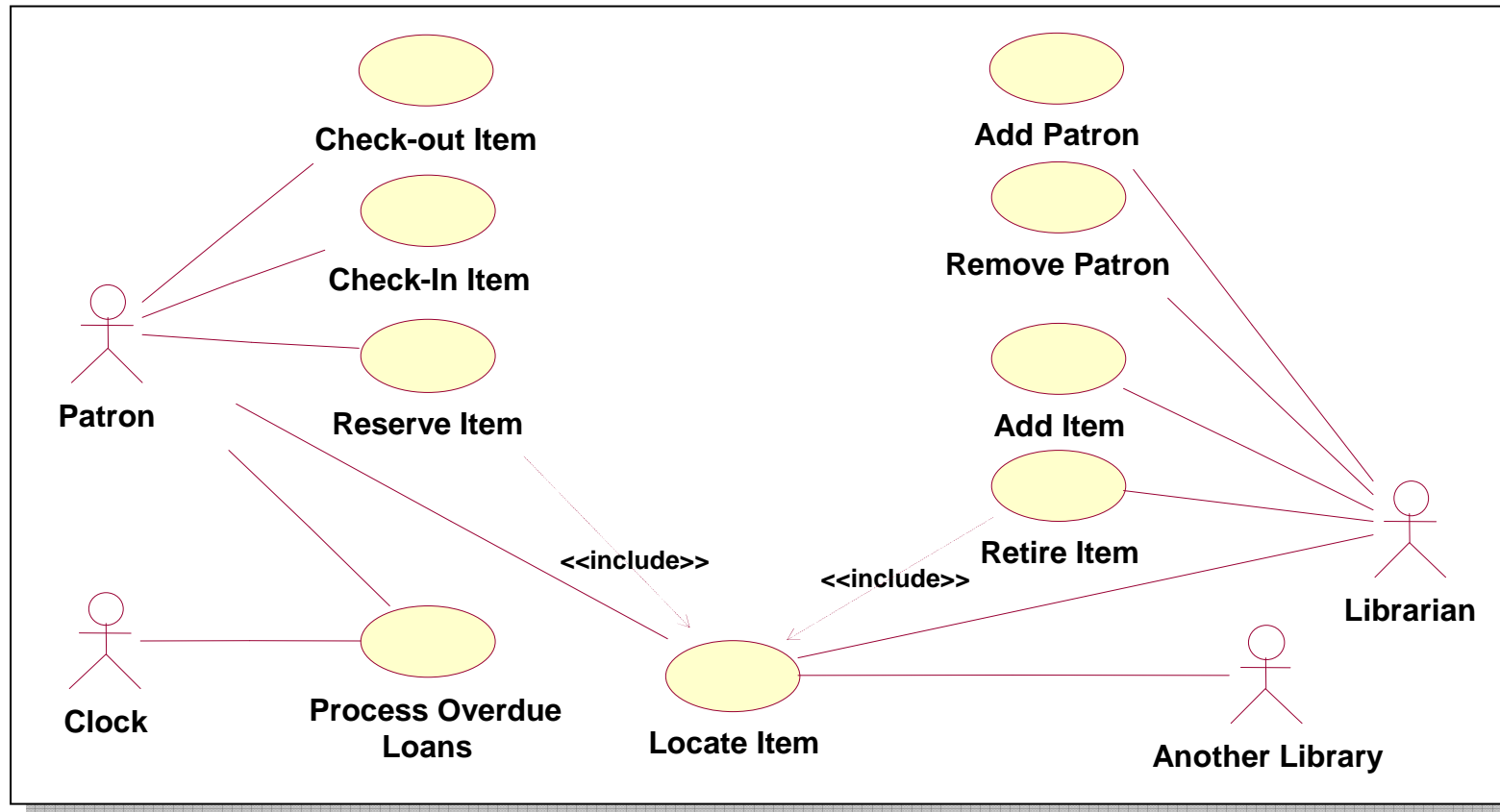
Use Case Model Defines Functional Scope

- A use case describes a use of the system in terms of the actor actions and corresponding systems responses that compose that use
 - A projects use case model usually has many use cases
 - A use case usually documents many scenarios, variations on the target use
- Actors are external roles that interact directly with the system
 - Human users
 - External systems



Example Use Case Diagram

Library Management System



Sample Use Case Specification (Part 1 Of 5)

Name: Checkout Item

Description:

Allow a patron to checkout from the library one or more items, which may be any combination of books, CDs, and periodicals.

Actors:

Patron

Goals:

The Patron wants to checkout one or more items.

Preconditions:

None

Post-Conditions:

(Flow A) Patron successfully checks-out one or more items.

(Flow B, D) Request is terminated without processing checkout.

Parent Use Case: None

Extended Use Cases: None

Included Use Cases: None

Sample Use Case Specification (Part 2 Of 5)

Basic Flow of Events:

A. Patron Checks-Out Item

1. The patron requests to checkout items and identifies himself or herself by scanning his or her library card with the barcode scanner.
2. The system verifies the identity of the patron and that the patron is in good standing and able to checkout items.

Rule: A patron is in good standing if his or her library membership has not expired, has no overdue items, and owes no past-due fees.

Rule: A patron may not have more than 10 items checked-out at any time.

[Alternate: Patron Unable To Checkout Items (B)]

3. The patron requests to checkout an item and identifies the item by scanning the item's identification label with the barcode scanner.

[Alternate: Item Identified Using Item Call Number (C)]

Sample Use Case Specification (Part 3 Of 5)

4. The system verifies that the item is available for checkout.
 - Rule: Reference materials and rare items may not be checked-out.
 - Rule: Items that are reserved by a patron other than this patron may not be checked-out.
 - [Alternate: Item Not Available For Checkout (D)]
5. The system records that the item is checked out to the patron and prints a date due slip on the printer.
6. The system asks the patron if there are any more items to process.
7. The patron indicates that no more items remain.
 - [Alternate: Patron Has More Items To Process (E)]
8. The use case ends.

Sample Use Case Specification (Part 4 of 5)

Alternate Flow of Events:

B. Patron Unable To Checkout Items

1. The system determines that the patron is not able to checkout items and displays a message indicating the reason to the patron.
2. The use case ends.

C. Item Identified Using Item Call Number

1. The patron identifies an item to checkout by entering its Call Number.
2. The use case resumes at A.4.

D. Item Not Available For Checkout

1. The system determines that the item is not available for checkout and displays a message indicating the reason to the patron.
2. The use case resumes at A.6.

Sample Use Case Specification (Part 5 Of 5)

E. Patron Has More Items To Process

1. The patron indicates that more items remain to checkout.
2. The system verifies that the patron may continue to checkout items.

Rule: A patron may not have more than 10 items checked-out at any time.

[Alternate: Patron Unable To Checkout Items (B)]

3. The use case resumes at A.3.

Other Requirements:

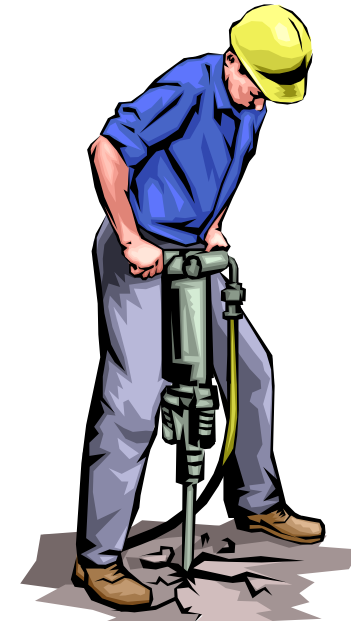
None

Use Cases & Risks

- **Software Architect is responsible for mapping risks from the project Risk List to each use case**
- **Determine priority of each risk by considering**
 - **Probability that the risk will occur**
 - **Impact of risk should it occur**
- **A risk level is determined for each use case**
 - **Which risks use case exposes project to**
 - **Degree to which realizing use case will mitigate each risk**
- **These factors determine the use case risk-based priority**
 - **Individual use case scenarios may be considered in the risk assessment**

Use Case Priority Drivers

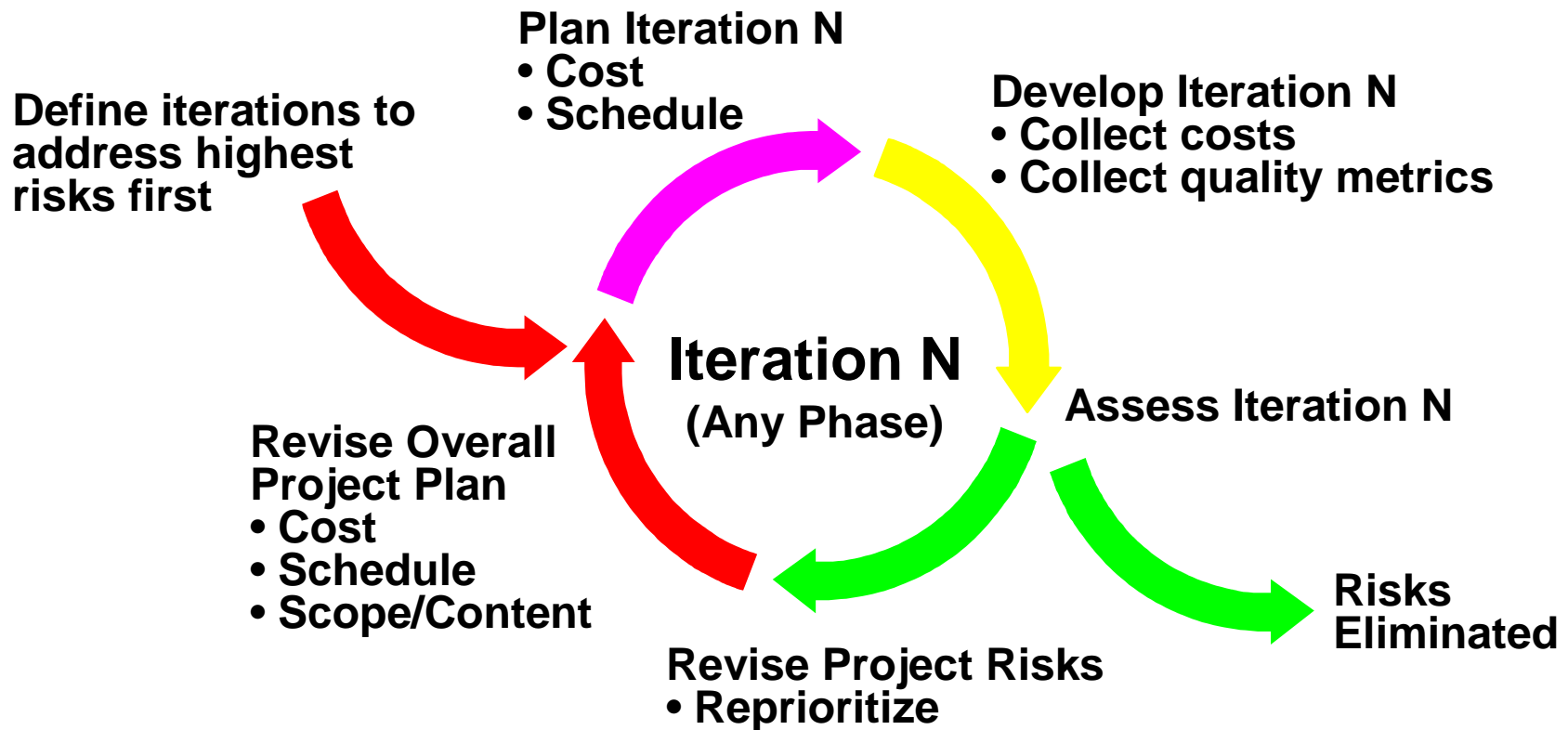
- Use case priority determines assignment of use cases to iterations
- Use case priority is driven by
 - Risk
 - Architectural significance
 - Architectural dependencies
 - Business priorities
- Iteration planning should ensure that
 - All architecturally significant use cases have been validated by end of Elaboration phase
 - All high and medium risks have been mitigated by Elaboration
 - All high business priorities have been met by early Construction, iterations with medium priority to follow
 - Other tactical objectives or constraints, such as demo to end-user, are met



Non-Functional Requirements

- **Non-functional requirements usually apply to several use cases or the entire system**
- **Some typical non-functional requirements include**
 - **Performance**
 - **Reliability**
 - **Supportability**
 - **Usability**
 - **Security**
- **These should be prioritized in addition to the use cases**
- **Many of the same priority drivers can be applied, such as**
 - **Architectural significance**
 - **Stakeholder importance**

Risk Reduction Drives Iterations



Risk-to-Use Case Matrix

<i>Risk Matrix</i>		Use Case Identifier													
Risk Priority	Risk Title	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	First Time Using Of Web Server Technology	X	X	X		X				X	X		X		
2	XML Interface to Banking Services			X					X						
3	Dependency on ACME Tax Wizard Enhancements		X												
4	Use of Java by Untrained Staff	X	X	X	X	X	X			X	X		X		X
5	Upgrade to Version 7.0 of DBMS			X	X			X							
6	Etc. ...														

- Architectural component to use case matrix can also be created

Simple Use Case Partitioning Results

- **Base use case partitioning on**
 - Complexity
 - Ease of Developing
 - Level of Risk
- **Other factors to consider**
 - Inter-use case dependencies
 - Stakeholder priorities

Use Case ID	Use Case Name	Complexity	Ease of Developing	Level of Risk	Partitioning Factor	Assigned Iteration
1	Find Mortgage Products	4	3	3	10	2
2	Compute Monthly Payments	2	2	2	6	3
3	Apply for Mortgage	5	4	5	14	1

- **Partitioning Factor = Complexity + Ease + Risk**
- **Higher Partitioning Factors are assigned to earlier iterations**

Determining Evaluation Criteria

- Describe the criteria to be used at the end of the Iteration for evaluating whether the iteration met its objectives

- Address the primary goals for the current Phase:
 - Inception Phase → Customer Acceptance, Proof-of-Concept
 - Elaboration Phase → Architecture Stability, Interface Stability
 - Construction Phase → Breakage, Defect Density, Discovery Rates
 - Transition Phase → Defect Density, End-User Acceptance

- Questions to address:
 - Did planned use cases get built?
 - Did planned architectural components get built?
 - Did planned risks get mitigated?
 - Were schedule and budget targets met?

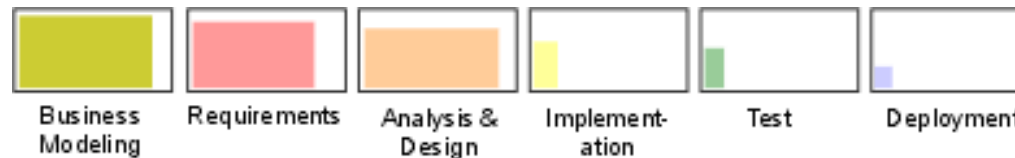
Defining Iteration Activities

- **Define activities to support the scope and goals of the iteration**
- **Activities should follow the process as described in the Development Case:**
 - **Select the Use Cases which exercise required functionality**
 - **Research and document the Use Case**
 - **Analyze the Use Case and allocate it to subsystems and classes**
 - **Design, implement, and unit-test classes and subsystems.**
 - **Integrate and test the system as a whole**
 - **Put the system in a releasable form**

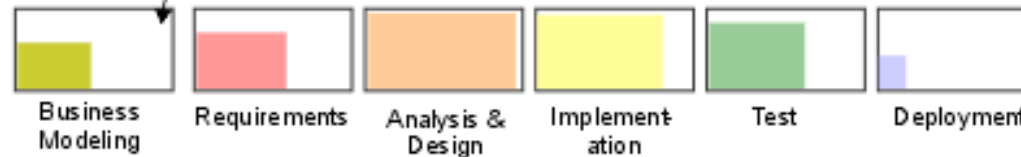
Iterative Content

- Each iteration executes all of the disciplines

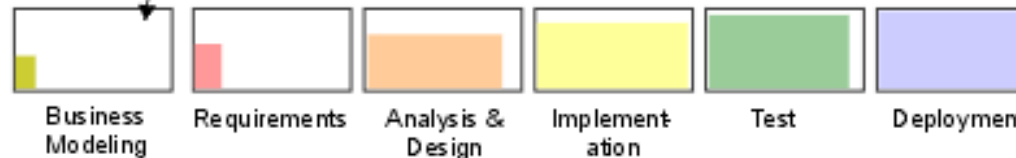
Elaboration
Iteration 1



Elaboration
Iteration 2



Construction
Iteration 1



Assign Responsibility

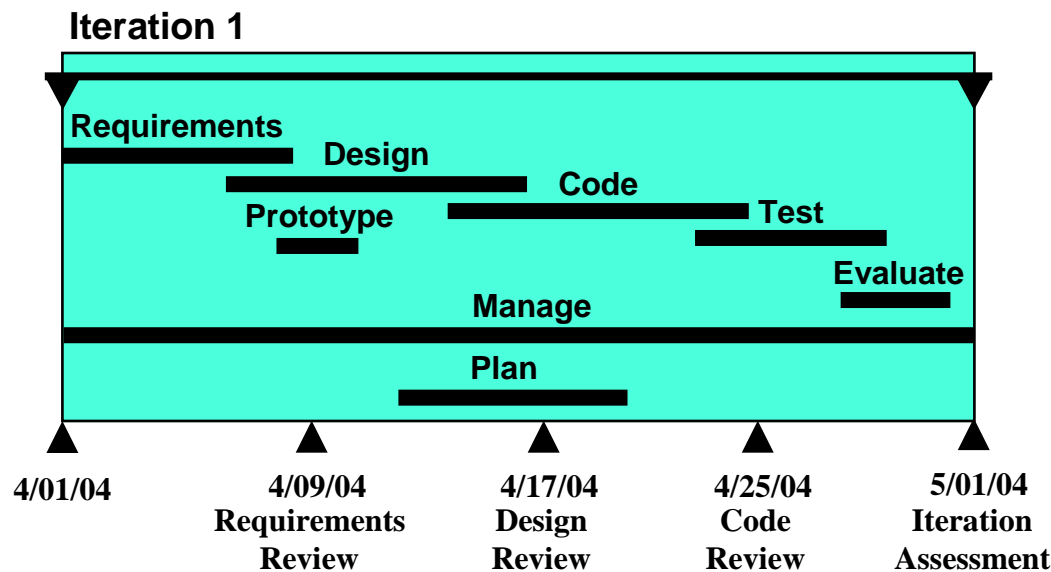
- Assign responsibility for each activity to an individual project team member. Activities may be carried out by an individual or a team.
- Appoint a single point of contact for each activity. That will help ensure consistency.

It's yours



Build a Schedule

- Use a tool to build a schedule for the iteration's activities and assigned resources (example: Microsoft Project)
- All tasks and artifacts should be included
- Timeline should fit within project plan



Iteration Strategies

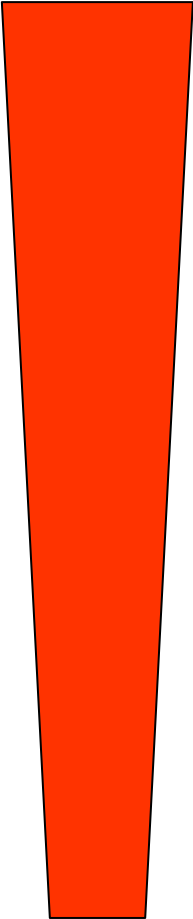
- **Iteration Strategy defines the depth and breadth of system development.**
- **There are two basic iteration strategies:**
 - **Wide and Shallow**
 - **Narrow and Deep**

Wide and Shallow Strategy



- **The Wide and Shallow Strategy is to implement a small portion of many requirements.**
- **This strategy is appropriate when:**
 - **The team is inexperienced in a technology, methodology, or process**
 - **Sound architecture is a key requirement for future capability, and the architecture is unprecedented.**
- **Potential Pitfall of this strategy include:**
 - **Analysis Paralysis**
 - **Team can lose confidence**
 - **The real technical risks can be hidden**

Narrow and Deep Strategy

- 
- **The Narrow and Deep strategy is to thoroughly implement a few requirements.**
 - **This strategy is appropriate when:**
 - **Early results need to be demonstrated to overcome a dominate risk, garner support, or prove viability**
 - **Requirements are continually evolving**
 - **The deadline is mandatory and an early start on development is key to success.**
 - **Potential pitfalls of this strategy include:**
 - **Software may be incompatible with subsequent development**
 - **Difficulty in developing an unpredictable architecture**

Hybrid Strategies

- **Early iterations will have more Wide/Shallow flavor**
- **Later iterations tend to follow Narrow/Deep strategy**
- **Example:**
 - **Narrow/Deep strategy used in Inception**
 - **Wide/Shallow strategy used in Elaboration**
 - **Narrow/Deep strategy used in Construction**

Benefits of Iterative Development

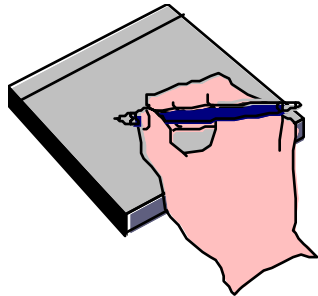
- **Earliest iterations mitigate highest risks**
- **Better use of different team roles over time**
- **Incorporates problems/issues/changes into future iterations**
- **Frequent, executable deliveries drive teams to get closure at regular intervals**
 - **Facilitates end user involvement early and often**
 - **Integration & test are continuous, not done in one lump just before delivery**
 - **Avoids the "90% done, 90% remaining" syndrome**
 - **Each release further mitigates risks**

Team Effort

- **Project planning using iterative development is a team effort**
 - This is hard work involving carefully considered judgments by many people
- **Project managers may be in charge but they rely upon project engineers, application architects, and other members of the team**
- **Project managers get help from others**
 - They are not lone wolves

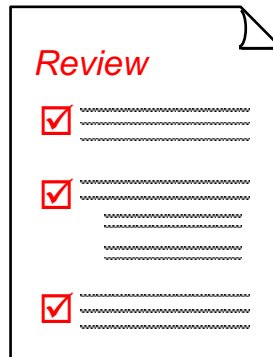


Lab 3: Iteration Planning



- **Demonstrate Filling Out An Iteration Program Plan**
- **Teams Plan Iteration E-1**
- **Teams Fill Out Iteration Plan**
- **Customer Assesses Iteration**
- **If necessary, perform another Elaboration iteration to achieve the milestones**
- **An Unexpected Event**
- **Revise estimate**

Lesson Review



- ✓ Iteration plans are fine grained plans of incremental development
- ✓ Iteration plans include
 - ✓ Goals
 - ✓ Use cases to develop
 - ✓ Risks to mitigate
 - ✓ Schedules
- ✓ Use cases partition system scope into manageable units that can be prioritized and assigned to iterations
- ✓ To prioritize use cases, we consider
 - ✓ Risk, Complexity, Ease, Dependencies, Business Need
- ✓ Non-functional requirements can drive priorities also