Object-Oriented Analysis and Design
Lecture 1: Best Practices of Software Engineering
Objectives

- Identify activities for understanding and solving software engineering problems.
- Explain the Six Best Practices.
- Present the Rational Unified Process (RUP) within the context of the Six Best Practices.
Content Outline

★ Software development problems

- The Six Best Practices
- RUP within the context of the Six Best Practices
Symptoms of Software Development Problems

✓ User or business needs not met
✓ Requirements not addressed
✓ Modules not integrating
✓ Difficulties with maintenance
✓ Late discovery of flaws
✓ Poor quality of end-user experience
✓ Poor performance under load
✓ No coordinated team effort
✓ Build-and-release issues
Trace Symptoms to Root Causes

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Root Causes</th>
<th>Best Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Needs not met</td>
<td>Insufficient requirements</td>
<td>Develop Iteratively</td>
</tr>
<tr>
<td>Requirements churn</td>
<td>Ambiguous communications</td>
<td>Manage Requirements</td>
</tr>
<tr>
<td>Modules do not fit</td>
<td>Brittle architectures</td>
<td>Use Component Architectures</td>
</tr>
<tr>
<td>Hard to maintain</td>
<td>Overwhelming complexity</td>
<td>Model Visually (UML)</td>
</tr>
<tr>
<td>Late discovery</td>
<td>Poor testing</td>
<td>Continuously Verify Quality</td>
</tr>
<tr>
<td>Poor quality</td>
<td>Subjective assessment</td>
<td>Manage Change</td>
</tr>
<tr>
<td>Poor performance</td>
<td>Waterfall development</td>
<td></td>
</tr>
<tr>
<td>Colliding developers</td>
<td>Uncontrolled change</td>
<td></td>
</tr>
<tr>
<td>Build-and-release</td>
<td>Insufficient automation</td>
<td></td>
</tr>
</tbody>
</table>
Content Outline

- Software development problems
- The Six Best Practices
- RUP within the context of the Six Best Practices
Practice 1: Develop Iteratively

Best Practices

*Process Made Practical*

Develop Iteratively
Manage Requirements
Use Component Architectures
Model Visually (UML)
Continuously Verify Quality
Manage Change
Waterfall Development Characteristics

- Delays confirmation of critical risk resolution
- Measures progress by assessing work products that are poor predictors of time-to-completion
- Delays and aggregates integration and testing
- Precludes early deployment
- Frequently results in major unplanned iterations

Waterfall Process

- Requirements analysis
- Design
- Code and unit test
- Subsystem integration
- System test
Iterative Development Produces an Executable

Each iteration results in an executable release
Risk Profiles

- Waterfall Risk
- Iterative Risk
- Risk Reduction

Object Oriented Analysis and Design
Practice 2: Manage Requirements

Best Practices

Process Made Practical

Develop Iteratively

Manage Requirements

Use Component Architectures

Model Visually (UML)

Continuously Verify Quality

Manage Change
Requirements Management

Making sure you
- solve the right problem
- build the right system

by taking a systematic approach to
- eliciting
- organizing
- documenting
- managing

the changing requirements of a software application.
Aspects of Requirements Management

- Analyze the Problem
- Understand User Needs
- Define the System
- Manage Scope
- Refine the System Definition
- Manage Changing Requirements
Map of the Territory

Problem

Needs

Features

Software Requirements

Test Scripts

Design

User Docs

Solution Space

The Product to Be Built

Traceability

Problem Space

Object Oriented Analysis and Design
Practice 3: Use Component Architectures

Best Practices

Process Made Practical

Develop Iteratively
Manage Requirements
Use Component Architectures
Model Visually (UML)
Continuously Verify Quality
Manage Change
Resilient Component-Based Architectures

- **Resilient**
  - Meets current and future requirements
  - Improves extensibility
  - Enables reuse
  - Encapsulates system dependencies

- **Component-based**
  - Reuse or customize components
  - Select from commercially available components
  - Evolve existing software incrementally
Purpose of a Component-Based Architecture

- **Basis for reuse**
  - Component reuse
  - Architecture reuse

- **Basis for project management**
  - Planning
  - Staffing
  - Delivery

- **Intellectual control**
  - Manage complexity
  - Maintain integrity

Component-based architecture with layers
Practice 4: Model Visually (UML)

Best Practices
Process Made Practical

Develop Iteratively
Manage Requirements
Use Component Architectures

Model Visually (UML)
Continuously Verify Quality
Manage Change
Why Model Visually?

- Captures structure and behavior
- Shows how system elements fit together
- Keeps design and implementation consistent
- Hides or exposes details as appropriate
- Promotes unambiguous communication
  - The UML provides one language for all practitioners
Visual Modeling With the Unified Modeling Language

- **Multiple views**
- **Precise syntax and semantics**

Diagram:

- Models
  - Dynamic Diagrams
    - Activity Diagrams
    - Statechart Diagrams
  - Static Diagrams
    - Use-Case Diagrams
    - Class Diagrams
  - Component Diagrams
    - Object Diagrams
    - Deployment Diagrams
  - Collaboration Diagrams
  - Sequence Diagrams
Visual Modeling Using UML Diagrams

Use-Case Diagram

Collaboration Diagram

Class Diagram

Statechart Diagram

Deployment Diagram

Component Diagram

Sequence Diagram

Forward and Reverse Engineering

Target System
Practice 5: Continuously Verify Quality

Best Practices

Process Made Practical

Develop Iteratively

Manage Requirements

Use Component Architectures

Model Visually (UML)

Continuously Verify Quality

Manage Change
Software problems are 100 to 1000 times more costly to find and repair after deployment

- Cost to Repair Software
- Cost of Lost Opportunities
- Cost of Lost Customers
Testing Dimensions of Quality

- **Usability**
  - Test application from the perspective of convenience to end user.

- **Functionality**
  - Test the accurate workings of each usage scenario.

- **Reliability**
  - Test that the application behaves consistently and predictably.

- **Supportability**
  - Test the ability to maintain and support application under production use.

- **Performance**
  - Test the online response under average and peak loading.
Test Each Iteration

UML Model and Implementation

Tests

Iteration 1

Iteration 2

Iteration 3

Iteration 4
Test Within the Product Development Lifecycle

Iteration $X$

- Requirements Capture
- Analysis and Design
- Project Planning
- Define Mission
- Validate Build
- Test and Evaluate
- Build
- Achieve Mission
- Improve Assets
- Verify Approach

Iteration $X + 1$

Iteration $X + 2$

Time
Practice 6: Manage Change

Best Practices
Process Made Practical

Develop Iteratively
Manage Requirements
Use Component Architectures
Model Visually (UML)
Continuously Verify Quality
Manage Change
What Do You Want to Control?

- Secure workspaces for each developer
- Automated integration/build management
- Parallel development
Aspects of a CM System

- Change Request Management (CRM)
- Configuration Status Reporting
- Configuration Management (CM)
- Change Tracking
- Version Selection
- Software Manufacture
Unified Change Management (UCM)

UCM involves:

- Management across the lifecycle
  - System
  - Project Management
- Activity-Based Management
  - Tasks
  - Defects
  - Enhancements
- Progress Tracking
  - Charts
  - Reports
Best Practices Reinforce Each Other

Best Practices

Develop Iteratively

Manage Requirements

Use Component Architectures

Model Visually (UML)

Continuously Verify Quality

Manage Change

Ensures users are involved as requirements evolve

Validates architectural decisions early on

Addresses complexity of design/implementation incrementally

Measures quality early and often

Evolves baselines incrementally
Module 1 Content Outline

- Software development problems
- The Six Best Practices

★✦ RUP within the context of the Six Best Practices
Rational Unified Process Implements Best Practices

Best Practices
**Process Made Practical**

- Develop Iteratively
- Manage Requirements
- Use Component Architectures
- Model Visually (UML)
- Continuously Verify Quality
- Manage Change
• Iterative approach
• Guidance for activities and artifacts
• Process focus on architecture
• Use cases that drive design and implementation
• Models that abstract the system
A process defines **Who** is doing **What**, **When**, and **How**, in order to reach a certain goal.

New or changed requirements → Software Engineering Process → New or changed system
Rational Unified Process has four phases:

- **Inception** - Define the scope of project
- **Elaboration** - Plan project, specify features and baseline architecture
- **Construction** - Build the product
- **Transition** - Transition the product into end-user community
Phase Boundaries Mark Major Milestones

Inception
- Lifecycle Objective Milestone (LCO)

Elaboration
- Lifecycle Architecture Milestone (LCA)

Construction
- Initial Operational Capability Milestone (IOC)

Transition
- Product Release
Iterations and Phases

An **iteration** is a distinct sequence of activities based on an established plan and evaluation criteria, resulting in an executable release (internal or external).

<table>
<thead>
<tr>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devel. Iteration</td>
<td>Devel. Iteration</td>
<td>Devel. Iteration</td>
<td>Transition Iteration</td>
</tr>
<tr>
<td>Transition Iteration</td>
<td>Transition Iteration</td>
<td>Transition Iteration</td>
<td></td>
</tr>
</tbody>
</table>

Minor Milestones: Releases
In an iteration, you walk through all disciplines.

Disciplines group activities logically.
Disciplines Produce Models

Disciplines

- Business Modeling
- Requirements
- Analysis & Design
- Implementation

Models

- Business Use-Case Model
- Use-Case Model
- Business Object Model
- Design Model
- Implementation Model

Realized By

Automated By

Implemented By

classes/code
Disciplines Guide Iterative Development

Business Modeling: Workflow

Requirements: Workflow
Overview of Rational Unified Process Concepts

- **Phase** → **Iteration** → **Discipline** → **Workflow**
- **Considers** → **Described by**
- **Role** Participates in **Activity** References **Artifact**
- **Responsible for** Modifies **Artifact**
- **Divided into** Considers **Detail**
- **Modifies** References **Model Element**
- **Participants in** Participates in **Role**
- **Document** Model **Artifact**

Object Oriented Analysis and Design
Review

- Best Practices guide software engineering by addressing root causes.
- Best Practices reinforce each other.
- Process guides a team on who does what, when, and how.
- The Rational Unified Process is a means of achieving Best Practices.