The Unified Software Process:
Best Practices for Software Development Teams
General Information

- Joel Pech, Software Engineering Specialist on Rational Software field team for IA-KS-NE & KC, MO
  jpech@rational.com, 319-228-8781
- Unified Software Development Process, by Ivar Jacobson, Grady Booch and Jim Rumbaugh
  Addison Wesley, 1999
- Rational Unified Process
  Our product that puts a unified software process on each desktop (some graphics for today’s talk are from this product).
Agenda

Why Rational Software focuses on process
- What is the problem
- 6 fundamental Best Practtices
- Structure of the Unified Software Process
- The Rational Unified Process product
Rational’s Mission

Ensuring the success of customers who depend on developing or deploying software
“Only 26% of software projects succeed.”

Standish Group, CHAOS Report, 1998
The Rules Have Changed

Release frequency

Cost of failure

Complexity

e-business
Enterprise
Web
Enterprise IT
ISV
Technical/Embedded

Accelerated Change

Rational
the e-development company
and, yet most tools are for ...

Individuals
The Answer: *Elevate the Solution*

**Teams**
Improve cross-functional *team* productivity

**Individuals**
Improve *individual* productivity
Rational Elevates the Solution

Organizations
Improve multi-team proficiency

Teams
Improve cross-functional team productivity

Individuals
Improve individual productivity
Agenda

- Why Rational Software focuses on process
- What is the problem
- 6 fundamental Best Practices
- Structure of the Unified Software Process
- The Rational Unified Process product
The Business Problem

Software:
- Increasingly more critical
- Needs to be delivered faster
- Needs to be higher quality

But...

“We estimate only 26% of software projects will succeed.”

Standish Group
CHAOS Report
1998
Rational’s Unique Approach

- Requirements Management
- Visual Modeling
- Automated Testing
- Change Management

Customer Success

- Tools
  - Develop Iteratively
  - Manage Requirements
  - Use Component Architectures
  - Model Visually (UML)
  - Verify Quality
  - Control Changes

- Services
  - Best Practices

- Customer Support
  - Tool Training
  - Project Implementation
  - Best Practices Education
  - Expert Consulting

Best Practices
Software Development is a Job for Teams

Challenges
- Larger teams
- Specialization
- Distribution
- Rapid technology change

Project Manager
Performance Engineer
Analyst
Developer
Tester
Release Engineer
Project
Manager
Rational
the e-development company
The Problem...

- Processes are not linked properly to tools, or are not properly automated.
- If process is used, different functional teams use normally inconsistent processes and modeling languages.
- Most software projects use no well-defined process. Instead team members (re-)invent process as they go.
Agenda

- Why Rational Software focuses on process
- What is the problem
- 6 fundamental Best Practices
- Structure of the Unified Software Process
- The Rational Unified Process product
6 Fundamental Best Practices

Capturing experiences from 1,000s of projects

- Develop Iteratively
- Manage Requirements
- Use Component Architectures
- Model Visually (UML)
- Verify Quality
- Control Changes
Best Practices Enable High-Performance Teams

Result

• More Successful projects

Best Practices

- Develop Iteratively
- Manage Requirements
- Use Component Architectures
- Model Visually
- Verify Quality
- Control Change

Rational
the e-development company
Practice 1: Develop Software Iteratively

- An initial design will likely be flawed with respect to its key requirements.
- Late-phase discovery of design defects results in costly over-runs and/or project cancellation.

The time and money spent implementing a faulty design are not recoverable.
Waterfall Development

- Requirements Analysis
- Design
- Code & Unit Testing
- Subsystem Testing
- System Testing

TIME
Waterfall Development: Risk vs. Time

- Requirements Analysis
- Design
- Code & Unit Testing
- Subsystem Testing
- System Testing
Each iteration results in an executable release.
Iterative Development Characteristics

- Use Case Driven
- Risk Driven
- Architecture Centric
Iterative Development Advantages

- Critical risks are resolved before making large investments
- Initial iterations enable early user feedback
- Testing and integration are continuous
- Objective milestones provide short-term focus
- Progress is measured by assessing implementations
- Partial implementations can be deployed
Practice 2: Manage Requirements

- Elicit, organize, and document required functionality and constraints
- Track changes and document tradeoffs and decisions
- Business requirements are easily captured and communicated through use cases
- Use cases are important planning instruments

Use Cases drives the work from analysis through test
Catch Requirements Errors Early

- Elicit stakeholder needs
- Gain agreement with the customer
- Analyze the problem
- Use an iterative approach
- Model system-user interaction
- Establish a baseline and change control process
- Maintain traceability and control of evolving requirements
Practice 3: Use Component Architectures

- Design, implement and test your architecture up-front!
- A systematic approach to define a “good” architecture
  - resilient to change by using well-defined interfaces
  - by using and reverse engineering components
  - derived from top rank use cases
  - intuitively understandable

Component-based Architecture with layers
Resilient, Component-Based Architectures

- **Resilient**
  - Meet current and future requirements
  - Improve extensibility
  - Enable reuse
  - Encapsulate system dependencies

- **Component-based**
  - Reuse or customization of components
  - Select from commercially-available components
  - Incremental evolution of existing software
Practice 4: Visually Model Software

- Capture structure and behavior of architectures and components
- Show how the elements fit together
- Hide or expose details as appropriate
- Maintain consistency between design and implementation
- Promote unambiguous communication

Visual modeling improves our ability to manage complexity
What Is the Unified Modeling Language?

- The UML is a language for
  - visualizing
  - specifying
  - constructing
  - documenting

the artifacts of a software-intensive system
Visual Modeling Using UML Diagrams

- Use case diagram
- Class diagram
- Statechart diagram
- Deployment diagram
- Collaboration diagram
- Component diagram
- Sequence diagram

Target System
Forward and Reverse Engineering
Practice 5: Verify Software Quality

- Create tests for each key scenario to ensure that all requirements are properly implemented
- Verify software reliability - memory leaks, bottle necks
- Unacceptable application performance hurts as much as missing functionality and unacceptable reliability
- Test every iteration - automate test!

Software problems are 100 to 1000 times more costly to find and repair after deployment
Testing in an Iterative Environment

Functionality

Iteration 1
Iteration 2
Iteration 3
Iteration 4

Test Suite 1
Test Suite 2
Test Suite 3
Test Suite 4

Automated Tests
Practice 6: Control Changes to Software

Change Request Management

Configuration Management

Measurement
Configuration Management Concepts

- Decompose the architecture into **subsystems**
- Assign subsystems to a **team**
- Establish **secure workspaces** for each developer
  - Provide isolation from changes made in other workspaces
  - Control all software artifacts - models, code, docs, etc.
- Establish an **integration** workspace
- Establish an enforceable **change control** mechanism
- Release a **tested baseline** at the completion of each iteration
Change requests come from many sources throughout the product lifecycle.

Change Request Management Concepts

Approved Decisions

Change Control Process

New Feature

New Requirement

Fix Bug

Vision

Require. & Use Cases

Code

Tests

Change Request System

Customer and End-User Inputs

Marketing

Developer inputs

Testers inputs
Measurement

Objective measurements provide meaningful project status information

Graph showing the number of closures and discovery per day, with a question about whether the trend is diverging or converging.
Best Practices Reinforce Each Other

Develop Iteratively

Manage Requirements

Use Component Architectures

Model Visually (UML)

Verify Quality

Control Changes

Ensures users involved as requirements evolve

Validates architectural decisions early on

Addresses complexity of design/implementation incrementally

Measures quality early and often

Evolves baselines incrementally
The result is software that is
- On Time
- On Budget
- Meets Users Needs

Best Practices
- Develop Iteratively
- Manage Requirements
- Use Component Architectures
- Model Visually
- Verify Quality
- Control Change
Agenda

- Why Rational Software focuses on process
- What is the problem
- 6 fundamental Best Practices

Structure of the Unified Software Process

- The Rational Unified Process product
Signifies a *role* that may be played by an individual or a team of individuals in the development organization.

**Worker**

**Use-Case Specifier**

**Activity**

**Artifact**

Describes a piece of work a worker may be asked to perform.

Signifies a piece of information that is produced, modified, or used by a process.

**Use Case**

**Use-Case Package**

**Describe a Use Case**
Workers Are Used for Resource Planning

Each individual in the project is assigned to one or several workers.
The Unified Software Process has four phases:

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

- **Purpose**
  - To establish the business case for a new system or for a major update of an existing system
  - To specify the project scope

- **Outcome**
  - A general vision of the project’s requirements, i.e., the core requirements
    - Initial use-case model and domain model (10-20% complete)
  - An initial business case, including:
    - Success criteria (e.g., revenue projection)
    - An initial risk assessment
    - An estimate of resources required

- **Milestone: Lifecycle Objectives**
Elaboration Phase

**Purpose**
- To analyze the problem domain
- To establish a sound architectural foundation
- To address the highest risk elements of the project
- To develop a comprehensive plan showing how the project will be completed

**Outcome**
- Use-case and domain model 80% complete
- An executable architecture and accompanying documentation
- A revised business case, incl. revised risk assessment
- A development plan for the overall project

**Milestone: Lifecycle Architecture**
Construction Phase

- **Purpose**
  - To incrementally develop a complete software product which is ready to transition into the user community

- **Products**
  - A complete use-case and design model
  - Executable releases of increasing functionality
  - User documentation
  - Deployment documentation
  - Evaluation criteria for each iteration
  - Release descriptions, including quality assurance results
  - Updated development plan

- **Milestone: Initial Operational Capability**
Transition Phase

- **Purpose**
  - To transition the software product into the user community

- **Products**
  - Executable releases
  - Updated system models
  - Evaluation criteria for each iteration
  - Release descriptions, including quality assurance results
  - Updated user manuals
  - Updated deployment documentation
  - “Post-mortem” analysis of project performance

- **Milestone: Product Release**
An *iteration* is a distinct sequence of activities with an established plan and evaluation criteria, resulting in an executable release (internal or external).
Iteration N Assessment

- Compare iteration actual cost, schedule, and content with iteration plan
- Determine rework (if any) to be done
  - Assign to future iteration(s)
- Determine what risks have been eliminated, reduced, or newly identified in this iteration
- Update project plan
- Prepare detailed plan for next iteration
  - Use revised risk list and select appropriate scenarios

Revised Risk List
- Total Cost
- Overall Schedule
- Scope/Content

Revised Project Plan
- Cost
- Schedule
- Content

Iteration N+1 Plan

Quality Assessment for Iteration N
- Test Results
- Defect Density
- Architecture Stability
- Other metrics

Iteration N Cost and Schedule Actuals
Each major workflow describes how to create and maintain a particular model.
Bringing It All Together...

Phases

- Process Workflows
  - Business Modeling
  - Requirements
  - Analysis & Design
  - Implementation
  - Test
  - Deployment

- Supporting Workflows
  - Configuration Mgmt
  - Management
  - Environment

In an iteration, you walk through all workflows.

Workflows group activities logically.

Iterations

<table>
<thead>
<tr>
<th>Preliminary Iteration(s)</th>
<th>Iter. #1</th>
<th>Iter. #2</th>
<th>Iter. #n</th>
<th>Iter. #n+1</th>
<th>Iter. #n+2</th>
<th>Iter. #m</th>
<th>Iter. #m+1</th>
</tr>
</thead>
</table>

Inception | Elaboration | Construction | Transition
Example of a Workflow

- System Analyst:
  - Develop Vision
  - Elicit Stakeholder Needs
  - Manage Dependencies
  - Capture a Common Vocabulary

- Use Case Author:
  - Find Actors and Use Cases
  - Structure the Use-Case Model
  - Detail a Use Case
  - User-Interface Modeling
  - User-Interface Prototyping

- User Interface Designer:
  - Prioritize Use Cases

- Architect:
  - Review Requirements

- Requirements Reviewer:
  - Requirements Review
Agenda

- Why Rational Software focuses on process
- What is the problem
- 6 fundamental Best Practices
- Structure of the Unified Software Process
- The Rational Unified Process product
Process Made Practical

- Role-based activity support
- Accessible, easy to use, web-enabled
- Automated by tools
- e-coach integrated with tools
- Configurable and customizable
Process Delivery In the Past….

- Thick binder on every developers shelf
- …collecting dust…
- hard to understand, hard to use, seen as driving overhead
Role-based Activity Support

- Value-add to each team member
- Detailed hands-on guidelines
- Productivity-enhancing document templates
- Non-intrusive

Worker

Use Case

Use-Case Package

Activity

Use-Case Specifier

Signifies a role that may be played by an individual or a team of individuals in the development organization

Describes a piece of work a worker may be asked to perform.

Artifact

Signifies a piece of information that is produced, modified, or used by a process

Describe a Use Case
Easy to Use

- Interactive knowledge base accessible from tools
- Web-enabled with powerful graphical navigation
- Search engine, index...

Searchable
Accessible
Navigable
Easy to use
Tool Mentor Topics - Using ClearCase to...

Set Up the Implementation Model
Create an Integration and Build Workspace
Create Private Workspace
Check Out and Check In
Configuration Items
Promote Configuration Items
Baseline the Project
Release Subsystem(s)
e-coach - Extended Help

Context sensitive process guidance from tools

Rational Rose - classics.mdl - [Class Diagram: Logical View / Top Level Arch]
Questions?

- Please feel free to contact me at jpech@rational.com or 319-228-8781
- www.rational.com
- IBM Application Framework for e-business
  - www.ibm.com/ebusiness
- Microsoft Windows DNA
  - www.microsoft.com/dna