The Rational Unified Process – Core Workflow Processes

- Requirements
- Analysis
- Design
- Implementation
- Test
The Unified Process as a Common Approach:

1. Provides guidance to the order of item’s activities
2. Directs the tasks of individual developers and the team
3. Specifies what artifacts should be developed
4. Offers criteria for monitoring and measuring the project’s products and activities

<table>
<thead>
<tr>
<th>The Triangle of Success</th>
<th>In SW Development</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Notation</strong></td>
<td>The Unified Modeling Language</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Process</th>
<th>Tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Rational Unified Process</td>
<td>Rational Rose CASE Tool</td>
</tr>
</tbody>
</table>
The Unified Process – History and Essence

Creating the Unified Process

Rational Unified Process 5.0
1998

Rational Objectory Process 4.1
1996-1997

Objectory Process 1.0-3.8
1987-1995

The Ericsson Approach

The Rational Approach

UML

Functional testing
Performance testing
Requirements mgmt
Conf. and change mgmt
Business engineering
Data engineering
UI design
The Process Itself

**What Is a Process?**

- Defines **Who is doing What**, **When to do it**, and **How to reach a certain goal**.

```
New or changed requirements    Software Engineering Process    New or changed system
```
The Unified Process is a Component Framework

There is NO Universal Process!
- The Unified Process is designed for flexibility and extensibility
  - allows a variety of lifecycle strategies
  - selects what artifacts to produce
  - defines activities and workers
  - models concepts
The Unified Process is Engineered

- **Use Case**: A use case is a description of a logic that can be carried out by the system.
- **Worker**: A role played by an individual or a team.
- **Analyst**: A role responsible for describing a use case.
- **Activity**: A unit of work.
- **Artifact**: A piece of information that is produced, modified, or used by a process.
- **Use Case Package**: A use case package represents a collection of use cases that are related to a specific functionality or domain.

**Description**

The Unified Process is a methodology for software development that is based on the rational unified process (RUP). It is designed to be a flexible and adaptable methodology that can be used for a wide range of software projects. The core workflow of the Unified Process includes several key elements:

1. **Use Case**: Use cases are descriptions of the functionality that the software is expected to provide. They are used to drive the development process and to ensure that the software meets the needs of its users.
2. **Worker**: Workers are the individuals or teams who carry out the activities in the process. They are responsible for producing and maintaining the artifacts that are created during the development process.
3. **Analyst**: The analyst is responsible for describing use cases. This involves understanding the requirements of the system and how they can be met by the software.
4. **Activity**: An activity is a unit of work that is performed by a worker. Activities are centered around a specific artifact and are designed to produce, modify, or use that artifact.
5. **Artifact**: An artifact is a piece of information that is produced, modified, or used by a process. Artifacts can include documents, models, code, and other types of software deliverables.

**Use Case Package**

A use case package is a collection of use cases that are related to a specific functionality or domain. It is used to organize and manage the use cases in a way that makes them easier to understand and work with.

**Worker Roles**

Workers play different roles in the Unified Process. These roles include:

- **Software Developer**: Responsible for implementing the software based on the design and requirements.
- **Quality Assurance Engineer**: Responsible for testing the software to ensure that it meets the requirements.
- **System Architect**: Responsible for designing the overall architecture of the system.
- **Project Manager**: Responsible for overseeing the project and ensuring that it is completed on time and within budget.

**Artifacts**

Artifacts are the deliverables that are produced, modified, or used by the process. They can include:

- **Requirements**: Descriptions of the requirements that the software must meet.
- **Design**: Descriptions of how the software will be implemented.
- **Code**: The actual code that implements the software.
- **Tests**: Tests that are used to verify that the software meets the requirements.

**Activity Types**

Activities can be categorized into different types based on their purpose. These types include:

- **Analysis**: Activities that focus on understanding the requirements and designing the solution.
- **Design**: Activities that focus on creating the detailed designs for the software.
- **Implementation**: Activities that focus on writing the code for the software.
- **Testing**: Activities that focus on verifying that the software meets the requirements.
- **Deployment**: Activities that focus on getting the software into production.

The Unified Process is designed to be a flexible and adaptable methodology that can be used for a wide range of software projects. It is based on a number of key principles, including:

- **Customer Focus**: The primary goal of the process is to meet the needs of the customer.
- **Flexibility**: The process is designed to be adaptable to the needs of the project.
- **Iterative Development**: The process is designed to be iterative, with multiple cycles of analysis, design, implementation, and testing.
- **Quality Assurance**: The process is designed to ensure that the software meets the requirements and is of high quality.

The Unified Process is a powerful methodology for software development that can be used to create software that meets the needs of its users.
The Unified Process is Architecture-Centric

- **Models are vehicles for visualizing, specifying, constructing, and documenting architecture**
- **The Unified Process prescribes the successive refinement of an executable architecture**

![Diagram showing the Unified Process phases: Inception, Elaboration, Construction, Transition. Time progresses from left to right.](Diagram.png)
Architecture and Models

Architecture embodies a collection of views of the models.
The Unified Process is Use-Case Driven

**Use Case Driven**

- **Use Cases bind these workflows together**

**Use Cases Drive Iterations**

- **Drive a number of development activities**
  - Creation and validation of the system's architecture
  - Definition of test cases and procedures
  - Planning of iterations
  - Creation of user documentation
  - Deployment of system

- **Synchronize the content of different models**
The Unified Process is Iterative and Incremental

**Lifecycle Phases**

- **Inception**: Define the scope of the project and develop business case
- **Elaboration**: Plan project, specify features, and baseline the architecture
- **Construction**: Build the product
- **Transition**: Transition the product to its users
An *iteration* is a sequence of activities with an established plan and evaluation criteria, resulting in an executable release.
The Unified Process in Software Development

Dependencies between use-case model and the other models
The 4 P’s in software development
Team working

- Workers and resources that realize them producing artifacts
Team workflow

A workflow with workers and activities

System Analyst

Architect

Use-Case Specifier

User-Interface Designer

Find Actors and Use Cases

Prioritize Use Cases

Structure the Use-Case Model

Detail a Use Case

Prototype User-Interface
Iterations and Core Workflows

Core Workflows

- Requirements
- Analysis
- Design
- Implementation
- Test

Phases

- Inception
- Elaboration
- Construction
- Transition

Iterations

- Preliminary Iteration(s)
- Iter. #1
- Iter. #2
- Iter. #n
- Iter. #n+1
- Iter. #n+2
- Iter. #m
- Iter. #m+1

An iteration in the elaboration phase
Requirements capture

From vision to requirements:
- List candidate requirements
- Understand system context
- Capture functional requirements
- Capture non-functional requirements

Use case example: Buyer, seller and payment handler are involved in the Sales – the payment handler transfers money from one account to another as specified by the invoice.
From requirements towards Use Cases

The workers and artifacts involved when capturing requirements as use cases

The inputs and outputs to/from the requirements capture
Analysis

Defines use-case realizations as structures of stereotypical classes and packages; gives structure to the internal view

The workers and artifacts involved in analysis

Alt. 1:
- Account
- Cashier Interface
- Withdrawal

Alt. 2:
- `entity`: Account
- `boundary`: Cashier Interface
- `control`: Withdrawal

UML provides standard stereotype classes for use in analysis
Analysis class diagrams

A class diagram for realization of the Pay Invoice use case

A collaboration diagram for realization of the Pay Invoice use case
The workers and artifacts involved in design.

Example: the invoice design class.
Design workflow

Architect

Use-Case Engineer

Component Engineer

Architectural Design

Design a Use Case

Design a Subsystem

Design a Class
Architectural design – inputs and outputs

- Use-Case Model
- Supplementary Requirements
- Analysis Model
- Architecture Description (view of the analysis model)
- Architect
- Subsystem (outlined)
- Interface (outlined)
- Design Class (outlined)
- Deployment Model (outlined)
- Architecture Description (views of the design and deployment models)
Implementation

From design to system components - source code, scripts, executables, etc.:
1. Plans the system iterations
2. System distribution by mapping components onto nodes in the deployment model
3. Implementation of design classes
4. Component test and integration

The workers and artifacts involved in implementation
Implementation – workflow and inputs/outputs
Implementation – system integration

The input and result of system integration

OOAD 4. The Rational Unified Process – Core Workflow
Processes
Test

Verification of result from implementation by testing each build (integration tests) and final releases (system tests).

1. Plans the tests required in each iteration
2. Test design and implementation:
   - test cases specifying the test;
   - test procedures – how to perform the test; exec.
   - test components for test automation

The workers and artifacts involved in testing.
Test

The workflow during testing.
The input and result of planning tests
The input and output of testing
Test example