



The Rap on RUP™ : An Introduction to the Rational Unified Process™

Jeff Jacobs

Jeffrey Jacobs & Associates

phone: 650.571.7092

email: jeff@jeffreyjacobs.com

<http://www.jeffreyjacobs.com>



Survey

- Does your organization have a well defined methodology/process?
- Does your organization use OOA/OOD?
- Does your organization use UML?



Agenda

- What is RUP?
- RUP Fundamentals
- Phases
- Product “features”
- Caveats
- Summary
- Questions



Process/Methodology Product Presentation

- Minimal UML bashing
- No rhyming
- No comparison with other methodologies
- RUP appears to be in flux since Rational's acquisition by IBM



Why RUP™ at SPIN?

- *The (RUP™) Knowledge base allows development teams to gain the full benefits of the industry standard UML*
- RUP™ covers all UML models
- RUP™ is **hot**; the latest silver bullet...



What is RUP™?

- A “software engineering process” (methodology)
- A knowledge base “process product”
 - CD to create web site
- UML model focused, not “paper documents” (but...)



What is RUP™?

- Configurable process/product
 - Recognizes and supports variety of different project types
 - Support for tailoring and configuring project web sites
- Project oriented

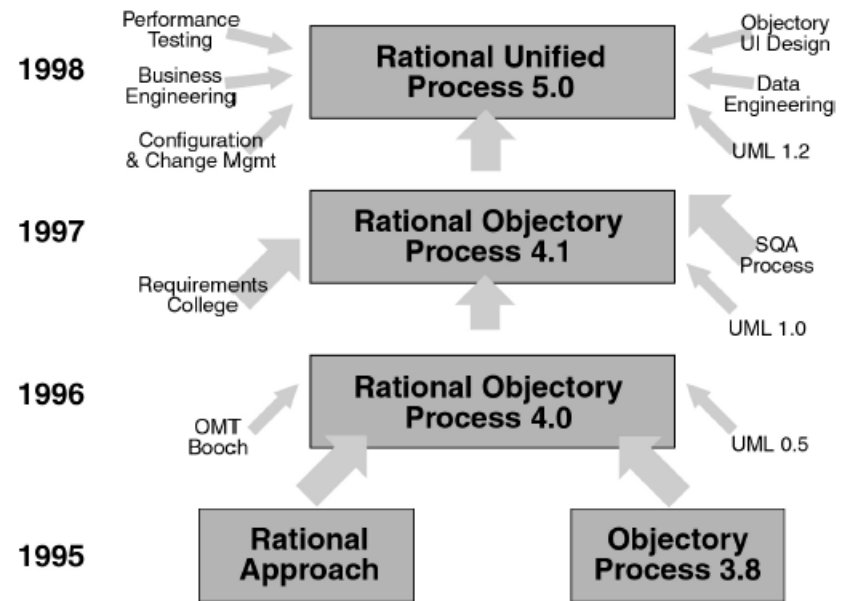


3 Flavors of RUP

- Generic - not dependant on specific technology
- Microsoft Web Solution Technology
 - Additional templates, guidelines etc
- IBM WebsphereTM Technology

History

- Methodology by Merger & Acquisition
- Objectory Process created in '80s
- Rational Approach created in '80s
- Acquisition of RequisitePro
- IBM acquires Rational



Genealogy of the Rational Unified Process



Why Should You Care About RUP™?

- Your organization is at SEI/CMM Level 1
“Ad Hoc”
 - Provides an excellent path to CMM Levels 2 and 3
- You need to add OOA/OOD to your current process/methodology
- Management wants to know when you’re going to use the latest silver bullet



RUP™ Fundamentals

- RUP is object and process oriented
 - Data takes a back seat
- Architecture is “key to success”
 - Emphasizes need for prototyping of core functionality, not just UI
- Iterative development
- Use Cases are (were?) primary requirements specification technique



4 Phases

- Inception
- Elaboration
- Construction
- Transition



Inception

- Establish business case and business models
- Establishes initial “vision”, high level requirements via “business” use cases
- Create stakeholder “buy in”
- Evaluate risks and return



Elaboration

- Detailed requirements
- Architecture and prototype
- Design



Construction

- Coding and testing



Transition

- Putting the product in the user's hands
- Highly variable, depending on product
 - Data migration
 - Training
 - Parallel Operations
 - Beta testing
 - Etc.

Overview of RUP™ (Organization)

Rational Unified Process: Overview

Disciplines

	Inception	Elaboration	Construction	Transition
Business Modeling	High	Medium	Low	None
Requirements	High	Medium	Low	None
Analysis & Design	Low	High	Medium	Low
Implementation	Low	Low	High	Medium
Test	Low	Low	High	Medium
Deployment	Low	Low	Low	High
Configuration & Change Mgmt	Low	Low	High	Medium
Project Management	Low	Low	High	Medium
Environment	Low	Low	Low	High

Iterations

Initial Elab #1 Elab #2 Const #1 Const #2 Const #N Tran #1 Tran #2

Click on an area of the screen for more information.

The Rational Unified Process® or RUP® product is a software engineering process. It provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end users within a predictable schedule and budget.

The preceding figure illustrates the overall architecture of the RUP, which has two dimensions:

- The horizontal axis represents time and shows the lifecycle aspects of the process as it unfolds. This first dimension illustrates the dynamic aspect of the process as it's enacted and is expressed



Best Practices

- Develop software iteratively
- Manage requirements
- Use component-based architectures
- Model visually
- Continuously verify quality
- Control changes



Key Concepts of RUP™

- Organized by *discipline*
- *Workflow* - model of process for a discipline
- *Workflow Details* - 2nd level detail of workflow, detailing activities, roles and artifacts
- *Role* - who performs an activity
- *Activity* - defined piece of work that results in an artifact



More Key Concepts

- *Artifact* - a deliverable, may be document, model, code, etc
 - Templates and examples for many artifacts
- *Tool Mentor* - guide on using Rational Tools for RUP™



Analyst Roles

- Business-Model Reviewer
- Business Designer
- Business-Process Analyst
- System Analyst
- Requirements Specifier
- Test Analyst
- User-Interface Designer



Developer Roles

- Capsule Designer
- Code Reviewer
- Database Designer
- Implementer
- Integrator



More Roles

- Testers
- Managers
- Process Engineer
- Project Manager
- Change Control Manager
- Configuration Manager
- Deployment Manager
- Project Reviewer
- Test Manager



Disciplines

- A collection of related activities that are related to a major 'area of concern' within the overall project
- Disciplines span phases



RUP™'s Disciplines

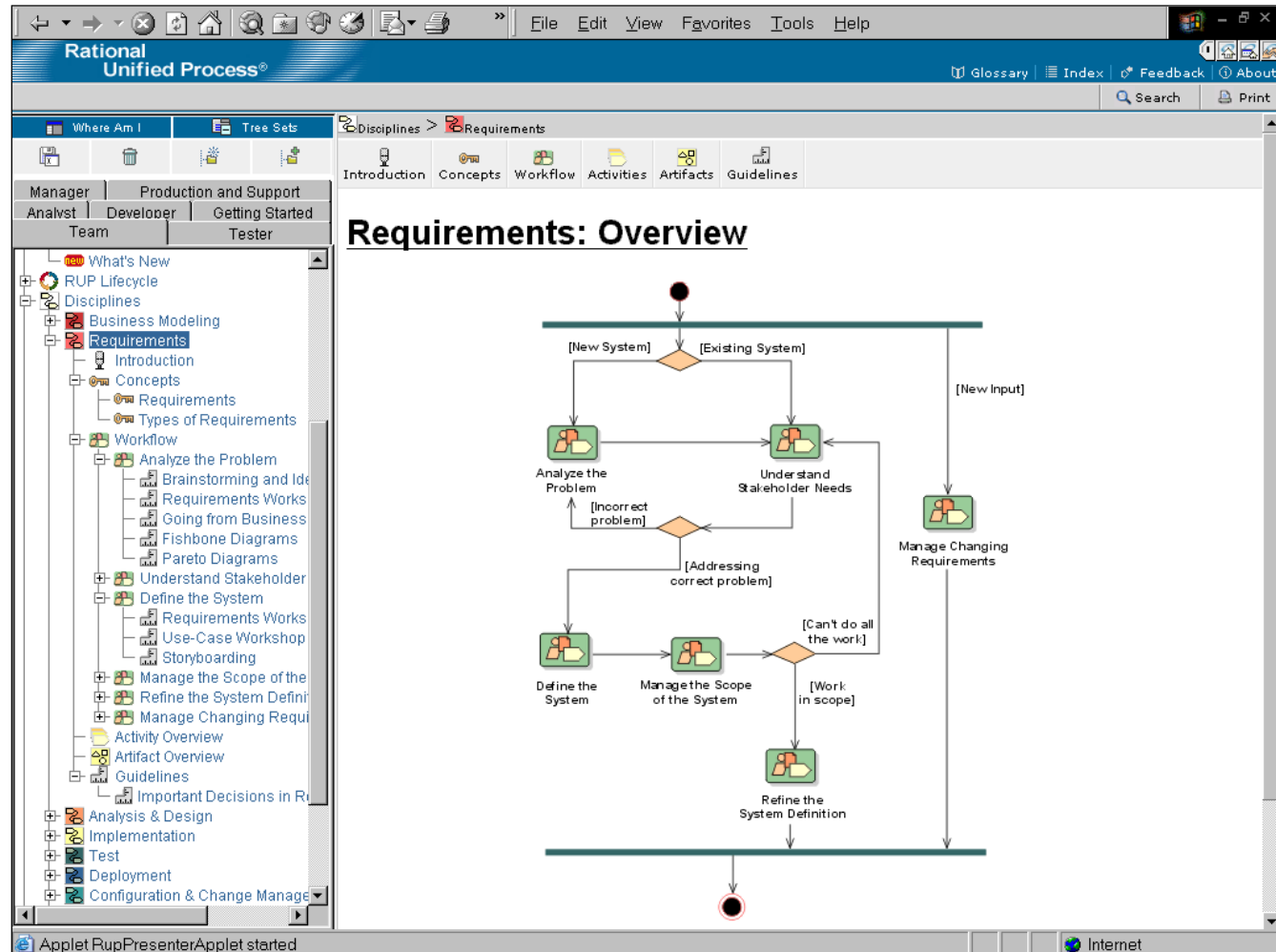
- Business Modeling
- Requirements
- Analysis and Design
(Analysis < >
Requirements, not
performed by “analyst”
role)
- Implementation
- Test
- Deployment
- Configuration and
Change Management
- Project Management
- Environment



Each Discipline is Composed of:

- Overview
- Introduction
- Concepts
- Workflow - the high level activity diagram (process flow)
- Workflow detail - second level process
- Activities - actions performed by roles
- Artifacts - deliverables
- Guidelines - tutorials, checklists, etc

Discipline Overview



Introduction

The screenshot displays the Rational Unified Process (RUP) software interface. The main window is titled 'Rational Unified Process' and shows a navigation pane on the left with a tree structure of topics. The 'Requirements' discipline is selected, and the 'Introduction' page is displayed in the main content area. The page title is 'Introduction to Requirements'. Below the title, there are two links: 'Purpose' and 'Relation to Other Disciplines'. The 'Purpose' section is expanded, showing a list of bullet points describing the purpose of the Requirements discipline. The text explains that the purpose is to establish and maintain agreement with customers and other stakeholders on what the system should do, to provide system developers with a better understanding of the system requirements, to define the boundaries of the system, to provide a basis for planning the technical contents of iterations, to provide a basis for estimating cost and time to develop the system, and to define a user-interface for the system, focusing on the needs and goals of the users. The text also mentions that to achieve these goals, it is important to understand the definition and scope of the problem, and that the Business Rules, Business Use-Case Model, and Business Analysis Model developed during Business Modeling will serve as valuable input to this effort. Stakeholders are identified and Stakeholder Requests are elicited, gathered and analyzed. A Vision document, a use-case model, use cases and Supplementary Specification are developed to fully describe the system - what the system will do - in an effort that views all stakeholders, including customers and potential users, as important sources of information (in addition to system requirements). Stakeholder Requests are both actively elicited and gathered from existing sources to get a "wish list" of what different stakeholders of the project (customers, users, product champions) expect or desire the

Introduction to Requirements

- [Purpose](#)
- [Relation to Other Disciplines](#)

Purpose

The purpose of the Requirements discipline is:

- To establish and maintain agreement with the customers and other stakeholders on what the system should do.
- To provide system developers with a better understanding of the system requirements.
- To define the boundaries of (delimit) the system.
- To provide a basis for planning the technical contents of iterations.
- To provide a basis for estimating cost and time to develop the system.
- To define a user-interface for the system, focusing on the needs and goals of the users.

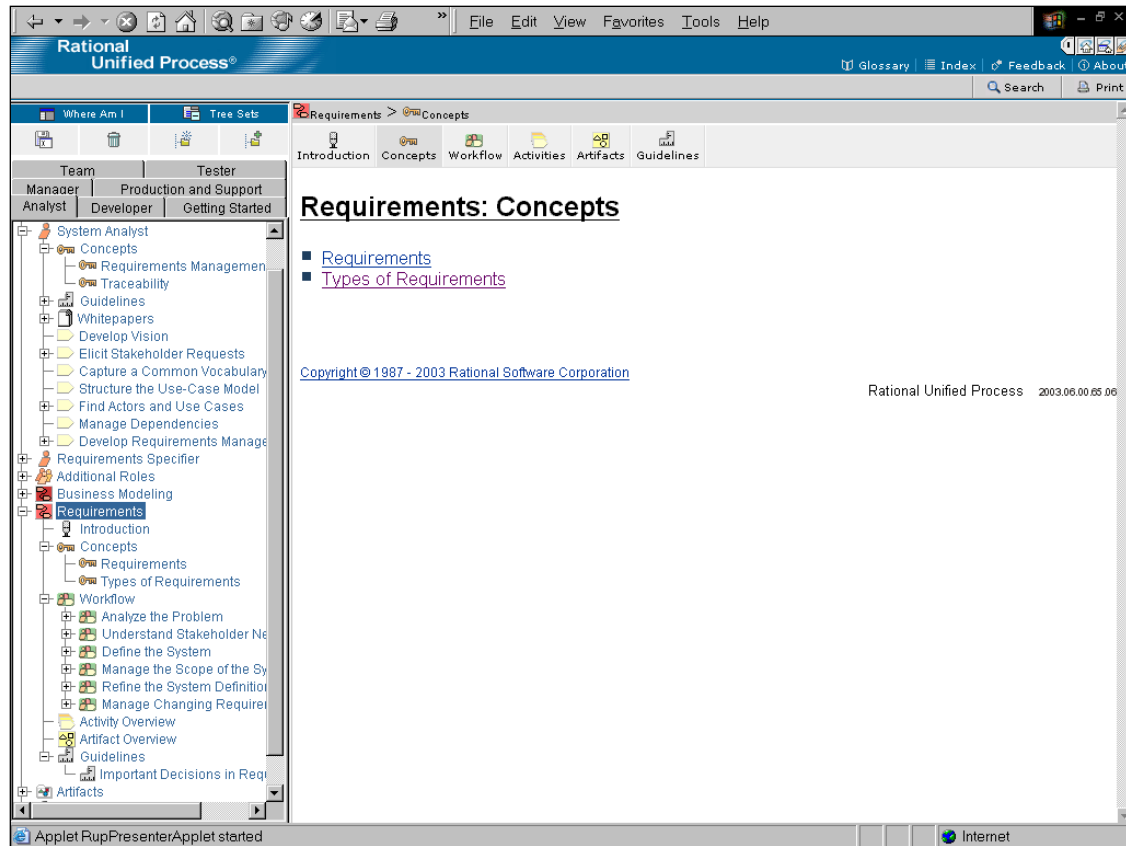
To achieve these goals, it is important, first of all, to understand the definition and scope of the problem which we are trying to solve with this system. The [Business Rules](#), [Business Use-Case Model](#) and [Business Analysis Model](#) developed during [Business Modeling](#) will serve as valuable input to this effort. [Stakeholders](#) are identified and [Stakeholder Requests](#) are elicited, gathered and analyzed.

A [Vision](#) document, a [use-case model](#), [use cases](#) and [Supplementary Specification](#) are developed to fully describe the system - **what** the system will do - in an effort that views all [stakeholders](#), including customers and potential users, as important sources of information (in addition to system requirements).

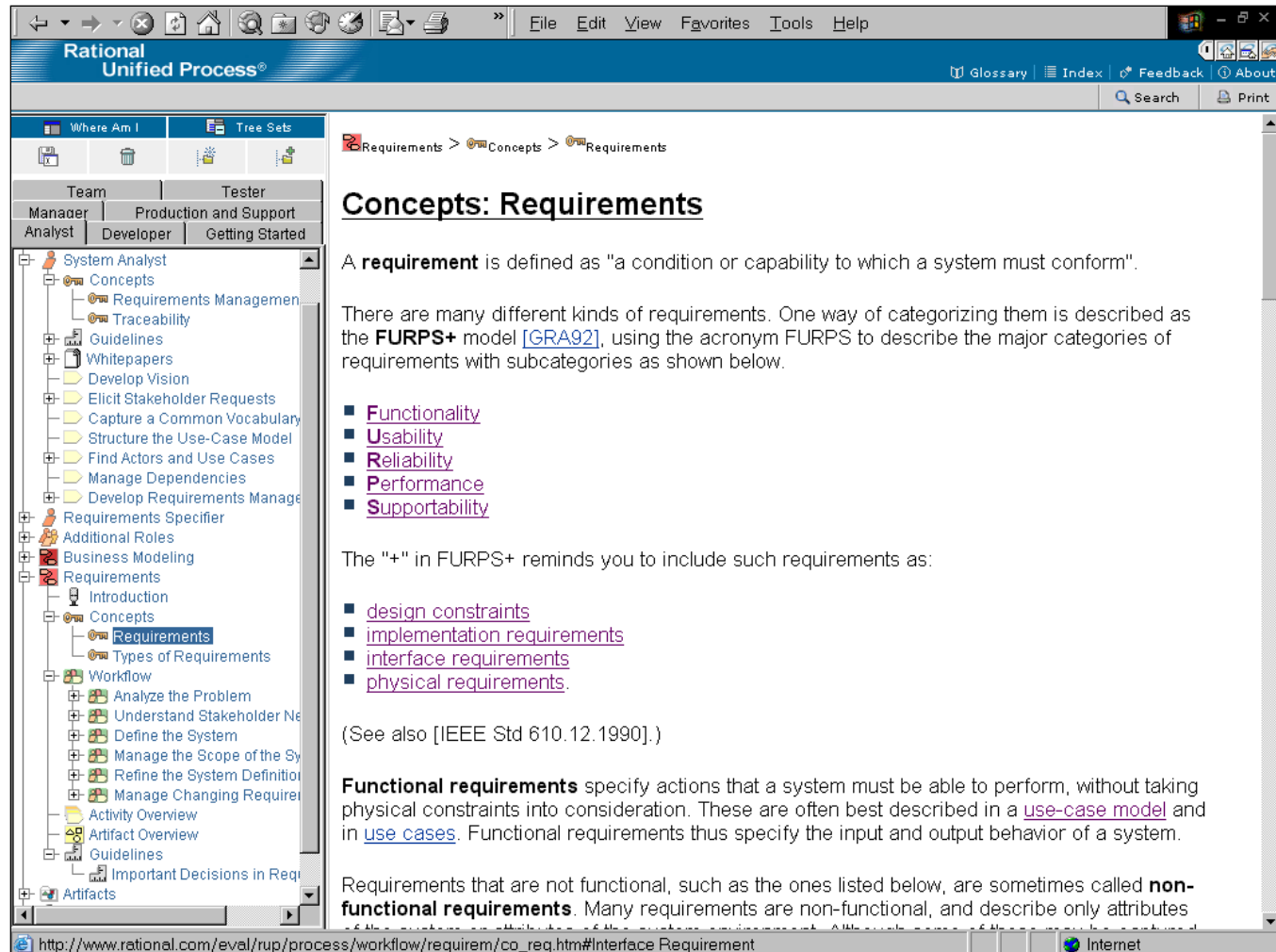
[Stakeholder Requests](#) are both actively elicited and gathered from existing sources to get a "wish list" of what different stakeholders of the project (customers, users, product champions) expect or desire the

Concepts

- Fundamentals of discipline/role



Concept Example



Rational Unified Process®

Where Am I | Tree Sets

Team: Manager, Analyst, Developer, Tester, Production and Support, Getting Started

Requirements > Concepts > Requirements

Concepts: Requirements

A **requirement** is defined as "a condition or capability to which a system must conform".

There are many different kinds of requirements. One way of categorizing them is described as the **FURPS+** model [GRA92], using the acronym FURPS to describe the major categories of requirements with subcategories as shown below.

- [Functionality](#)
- [Usability](#)
- [Reliability](#)
- [Performance](#)
- [Supportability](#)

The "+" in FURPS+ reminds you to include such requirements as:

- [design constraints](#)
- [implementation requirements](#)
- [interface requirements](#)
- [physical requirements](#)

(See also [IEEE Std 610.12.1990].)

Functional requirements specify actions that a system must be able to perform, without taking physical constraints into consideration. These are often best described in a [use-case model](#) and in [use cases](#). Functional requirements thus specify the input and output behavior of a system.

Requirements that are not functional, such as the ones listed below, are sometimes called **non-functional requirements**. Many requirements are non-functional, and describe only attributes

http://www.rational.com/eval/rup/process/workflow/requirem/co_req.htm#Interface Requirement

Everything is Use-Case Driven

Rational Unified Process®

File Edit View Favorites Tools Help

Glossary Index Feedback About

Search Print

Where Am I? Tree Set

Team: Manager, Analyst, Developer, Tester, Production and Support, Getting Started

Artifacts > Requirements Artifact Set > Use-Case Model > Use-Case View

Concepts: Use-Case View

To provide a basis for planning the technical contents of iterations, an [architectural view](#) called the **use-case view** is used in the [Requirements](#) discipline. There is only one use-case view of the system, which illustrates the use cases and scenarios that encompass architecturally significant behavior, classes, or technical risks. The use-case view is refined and considered initially in each iteration.

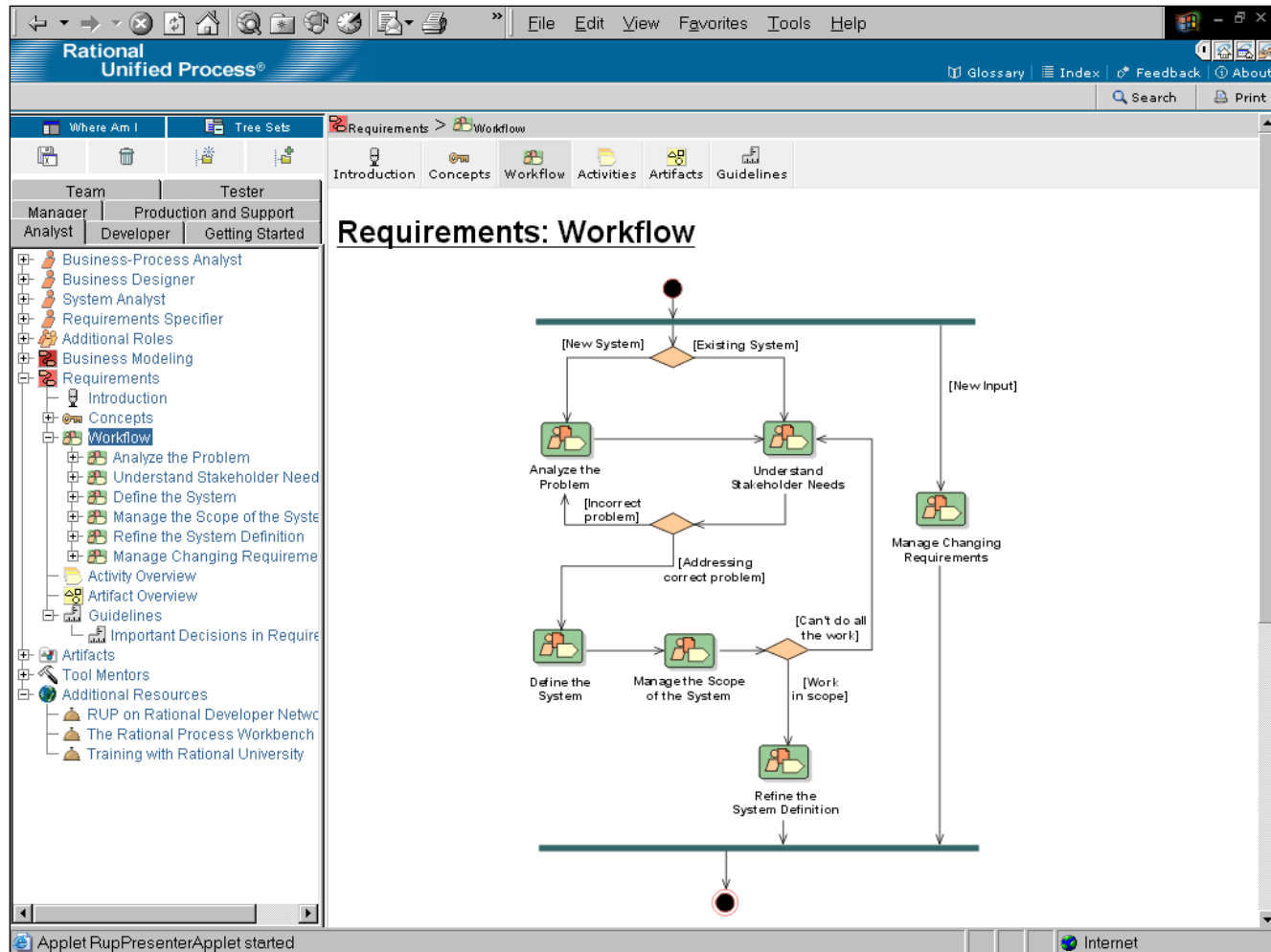
The Use-Case View shows an architecturally significant subset of the use-case model, a subset of the use cases and actors.

The analysis, design, and implementation activities subsequent to requirements are centered on the notion of an [architecture](#). The production and validation of that architecture is the main focus of the early iterations, especially during the [Elaboration](#) phase. Architecture is represented by a number of different [architectural views](#), which in their essence are extracts illustrating the "architecturally significant" elements of the models.

Applet RupPresenterApplet started

Internet

Workflow



Workflow Detail

Rational Unified Process®

File Edit View Favorites Tools Help

Glossary Index Feedback About

Search Print

Where Am I Tree Sets

Production and Support Team Tester
Analyst Developer Getting Started Manager

Business-Process Analyst
Business Designer
System Analyst
Requirements Specifier
Additional Roles
Business Modeling
Requirements
Introduction
Concepts
Workflow
Analyze the Problem
Brainstorming and Idea Reduction
Requirements Workshop
Going from Business Models to S
Fishbone Diagrams
Pareto Diagrams
Understand Stakeholder Needs
Define the System
Manage the Scope of the System
Refine the System Definition
Manage Changing Requirements
Activity Overview
Artifact Overview
Guidelines
Important Decisions in Requirements
Artifacts
Tool Mentors
Additional Resources
RUP on Rational Developer Network
The Rational Process Workbench Product
Training with Rational University

Requirements > Workflow > Analyze the Problem

Workflow Detail: Analyze the Problem

The purpose of this workflow detail is to gain agreement on the problem being solved. Analysis of the problem involves identify the stakeholders, define the boundary of, and identify the constraints imposed on the system.

Topics

- Description
- Related Information
- Timing
- Optionality
- How to
- Staff
- Work
- Guidelines

Description

The first step in any problem analysis is to make sure that all parties involved agree on what the problem is that needs to be solved—or opportunity that will be realized—by the system.

Applet RupPresenterApplet started

Internet

Activity

The screenshot displays the Rational Unified Process (RUP) web application interface. The top navigation bar includes links for Glossary, Index, Feedback, and About. The left sidebar shows a hierarchical tree of RUP artifacts, with 'Capture a Common Vocabulary' selected under the 'System Analyst' role. The main content area is titled 'Activity: Capture a Common Vocabulary' and contains the following sections:

- Purpose:**
 - To define a common vocabulary that can be used in all textual descriptions of the system, especially in use-case descriptions.
- Role:** [System Analyst](#)
- Steps:**
 - [Find Common Terms](#)
 - [Evaluate Your Results](#)
- Input Artifacts:**
 - [Business Analysis Model](#)
 - [Business Case](#)
 - [Business Rule](#)
 - [Business Use Case Model](#)
 - [Stakeholder Requests](#)
 - [Use Case](#)
 - [Use-Case Model](#)
 - [Vision](#)
- Resulting Artifacts:**
 - [Glossary](#)
- Tool Mentors:**
 - [Capturing a Common Vocabulary Using Rational RequisitePro](#)

The bottom status bar indicates 'AppletRupPresenterApplet started' and 'Internet' connectivity.

Activity Step

The screenshot displays the Rational Unified Process (RUP) software interface. The title bar reads 'Rational Unified Process®'. The menu bar includes 'File', 'Edit', 'View', 'Favorites', 'Tools', and 'Help'. The toolbar contains icons for navigation and editing. The main window is divided into two panes. The left pane, titled 'Where Am I', shows a hierarchical tree of project components. The right pane, titled 'Find Common Terms', contains text and a list of concepts.

Find Common Terms

In the Requirements discipline, you must define a common vocabulary using the most common terms in the problem domain. You should then consistently use the common vocabulary in all textual descriptions of the system, especially in use-case descriptions. In this way, you keep the textual descriptions consistent and avoid misunderstandings among project members about the use and meaning of terms. You should document the vocabulary in a glossary.

To find common terms in the problem domain, consider terms used in the requirements and the development team's general knowledge of the system to be built. Focus on terms describing the following concepts:

- Business objects representing concepts used in the organization's daily work or in the system's expected operating environment. In many cases, a list of concepts of this kind already exists.
- Real-world objects that the system needs to be aware of. These objects occur naturally, and include such things as: car, dog, bottle, aircraft, passenger, reservation, or invoice.

Example:

In a Depot-Handling System, conversation is about, among other things, the items in the depot and potential storage locations for them.


- Events that the system needs to be aware of. By "event" is here meant a point in time or a chronological incident that the system must know of, such as a meeting, or an error occurring.

Example:

A natural event in a Depot-Handling System is the delivery of goods to the depot. For each delivery the system should "remember" the date of the delivery, who received the goods, what goods were delivered and how many there are of each kind.

Artifacts May Be Documents, Models, Code, Etc.

The screenshot displays the Rational Unified Process (RUP) web application. The interface includes a top navigation bar with links for Glossary, Index, Feedback, and About. A left sidebar shows a tree view of project artifacts, with 'Software Requirements' and 'Use-Case Model' expanded. The main content area displays the details for the 'Glossary' artifact, which is represented by a document icon. The details are organized into a table-like structure with sections for Role, Optionality/Occurrence, Templates and Reports, Examples, UML Representation, More Information, Purpose, Timing, Responsibility, Tailoring, Input to Activities, and Output from Activities.

	The Glossary defines important terms used by the project.	
Role:	System Analyst	
Optionality/Occurrence:	Primary artifact used to capture information about the project's business domain. Inception and Elaboration phases.	
Templates and Reports:	■ Template: Glossary	
Examples:	■ CREG Glossary - Elaboration Phase ■ CREG Glossary - Inception Phase ■ CSPS Glossary V1.0 ■ CSPS Glossary V2.0	
UML Representation:	Not applicable.	
More Information:	■ Checklist: Glossary	
Purpose ■ Timing ■ Responsibility ■ Tailoring		
Input to Activities:	Architectural Analysis Assess Viability of Architectural Proof-of-Concept Detail a Use Case Detail the Software Requirements Find Actors and Use Cases	Output from Activities: ■ Capture a Common Vocabulary



Document Templates

- Templates for document artifacts available in a variety of formats
 - Microsoft Word
 - HTML
 - Framemaker
 - Rational SODA
- Business Glossary Template

Guidelines

Rational Unified Process®

Glossary | Index | Feedback | About

Search | Print

Where Am I | Tree Sets

Team | Tester

Manager | Production and Support

Analyst | Developer | Getting Started

Business-Process Analyst

Business Designer

System Analyst

Concepts

Requirements Management

Traceability

Guidelines

Whitepapers

Develop Vision

Elicit Stakeholder Requests

Capture a Common Vocabulary

Structure the Use-Case Model

Find Actors and Use Cases

Manage Dependencies

Develop Requirements Management

Requirements Specifier

Additional Roles

Business Modeling

Requirements

Introduction

Concepts

Workflow

Activity Overview

Artifact Overview

Guidelines

Important Decisions in Requirements

Artifacts

Tool Mentors

Additional Resources

RUP on Rational Developer Network

The Rational Process Workbench

Training with Rational University

Guidelines: Important Decisions in Requirements

Topics

- [Decide How to Perform the Workflow](#)
- [Decide How to Use Artifacts](#)
- [Decide Which Reports to Use](#)
- [Decide How to Maintain "Input Requirements"](#)
- [Decide How to Approve Use Cases](#)

Decide How to Perform the Workflow ⓘ

The following decisions should be made regarding the Requirements discipline's workflow:

- Decide how to perform the workflow by looking at the [Requirements: Workflow](#). Study the diagram with its [guard conditions](#). Decide which workflow details to perform and in which order.
- Decide what parts of the Requirements workflow details to perform. The table below shows some parts that can be introduced relatively independently from each other.
- Decide when, during the project lifecycle, to introduce each part of the workflow. As a general rule, the Requirements discipline should be introduced early in the project.

Part of workflow	Comments
Use-Cases	Some projects do not employ use-cases, which means that the project will not develop artifacts such as a Use-Case Model, Use-Case Package and Use Case. Instead use the Software Requirements Specification.
Workflow Detail: Manage Changing Requirements	This can be introduced after a few iterations in the project when there is a stable baseline.

AppletRupPresenterApplet started

Internet

Guideline Example

Rational Unified Process®

File Edit View Favorites Tools Help

Glossary Index Feedback About

Search Print

Where Am I Tree Sets

Manager Production and Support
Analyst Developer Getting Started
Team Tester

Project Management
Environment
Roles and Activities
Artifacts
Requirements Artifact Set
Software Requirement
Vision
Glossary
Stakeholder Requests
Storyboard
Software Requirements Specification
Supplementary Specification
Use-Case Model
Use-Case Model
Guidelines
Use-Case View
Use-Case Package
Use Case
Use Case guidelines
Use Case
Activity Diagram in the Use-Case Model
Use Case <use-case n
Templates
Examples
Actor
Use-Case Model Survey
Examples
Requirements Management
Requirements Attributes
Analysis & Design Artifact Set
Implementation Artifact Set
Test Artifact Set

Artifacts > Requirements Artifact Set > Use-Case Model > Use Case > Guidelines > Activity Diagram in the Use-Case Model

Guidelines: Activity Diagram in the Use-Case Model

The flow of events of a use case describes what needs to be done by the system to provide value to an actor. It consists of a sequence of activities that together produce something for the actor. The flow of events consists of a basic flow, and one or several alternative flows.

The flow of events of a use case can be described graphically with the help of an activity diagram. Such a diagram shows:

- **Activity states**, which represent the performance of an activity or step within the flow of events.
- **Transitions** that show what activity state follows after another. This type of transition is sometimes referred to as a completion transition, since it differs from a transition in that it does not require an explicit trigger event, it is triggered by the completion of the activity the activity state represents.
- **Decisions** for which a set of **guard conditions** are defined. These guard conditions control which transition (of a set of alternative transitions) follows once the activity has been completed. Decisions and guard conditions allow you to show **alternative threads** in the flow of events of a use case.
- **Synchronization bars** which you can use to show parallel subflows. Synchronization bars allow you to show **concurrent threads** in the flow of events of a use case.

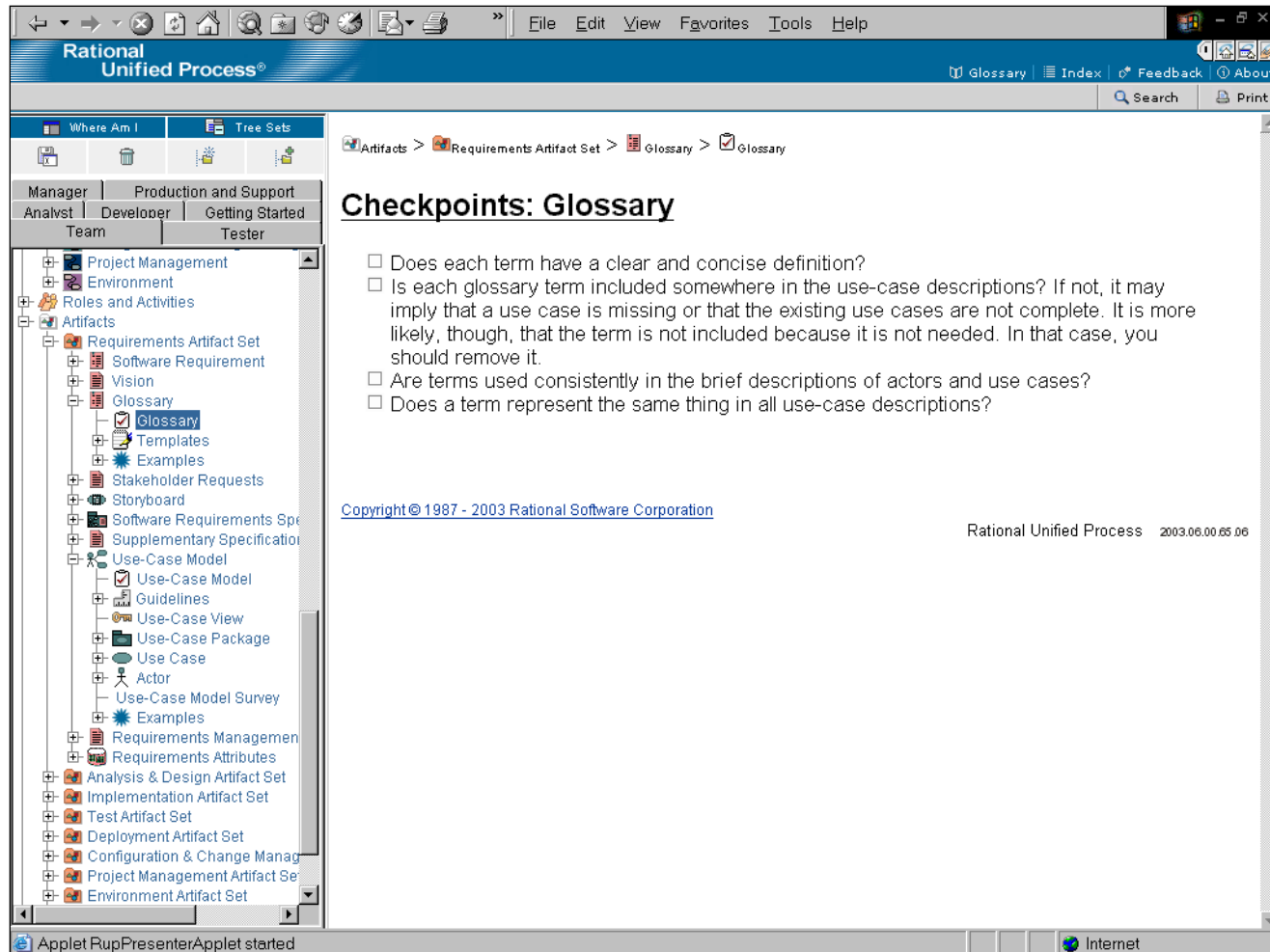
Activity state

Alternative threads

Guard condition

```
graph TD
    Start(( )) --> Verify[Verify access code]
    Verify --> Decision{ }
    Decision -- "[ incorrect ]" --> Handle[Handle incorrect access code]
    Decision -- "[ correct ]" --> End(( ))
```

Checkpoints for Quality Reviews





More Stuff

- Sample Projects
- Project Management Templates
- Tool Mentors



Sample Projects

- Examples of many artifacts for two projects
 - Course Registration System
 - Collegiate Sports Paging Systems

Glossary Example

The screenshot shows the Rational Unified Process (RUP) software interface. On the left is a project tree with categories like Team, Tester, Manager, and Analyst. The main window is titled 'Revision History' and contains a table with the following data:

Date	Version	Description	Author
26/Dec/1998	1.0	Draft version	Bill Collings
19/Feb/1999	2.0	Expand glossary. Moved some of the terms to the Wylie College glossary.	Bill Collings

Below the table is a section titled 'Glossary' with two numbered sections:

- 1. Introduction**

The glossary contains the working definitions for terms and classes in the Course Registration System. This glossary will be expanded throughout the life of the project. Any definitions not included in this document may be included in the Rational Rose Model. Generic terms used outside this project should be captured in the organizational Glossary.
- 2. Definitions**
 - Alternative course selection**

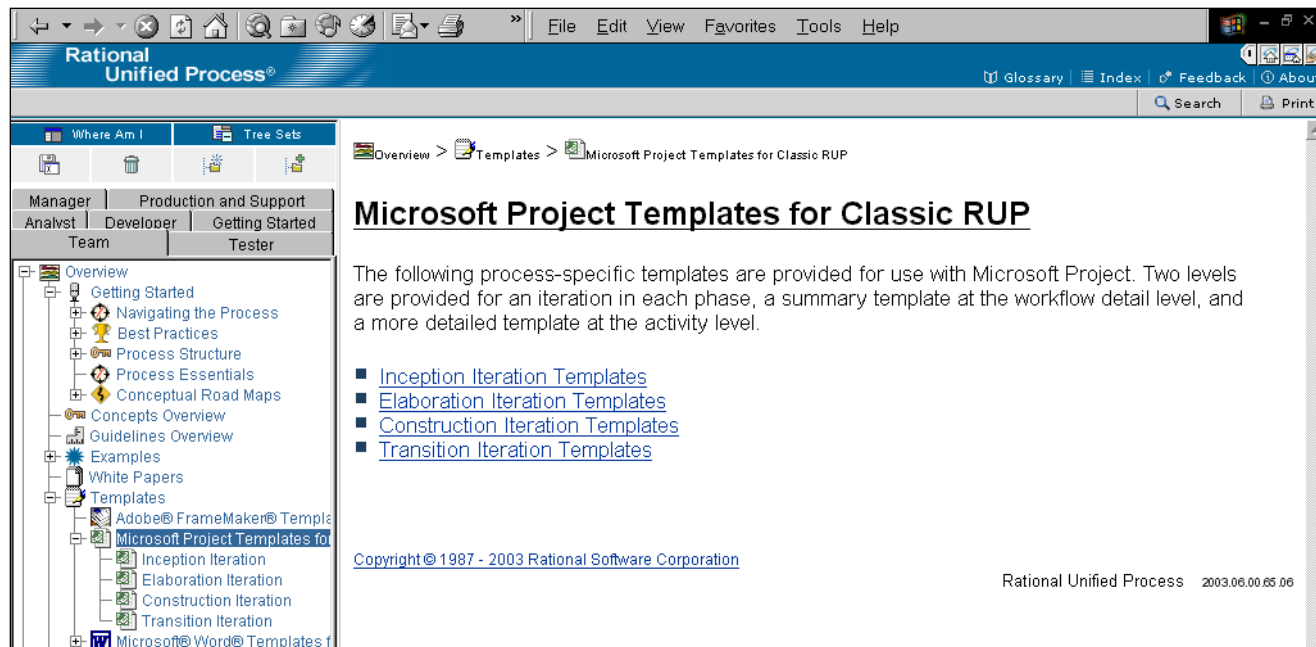
A student might choose to register for one or more alternative courses, in case one or more of the primary selections are not available.
 - Billing System**

Part of the university's Finance System used for processing billing information.
 - Prerequisite**

The university requires for some courses that a student has passed one or more other courses to be

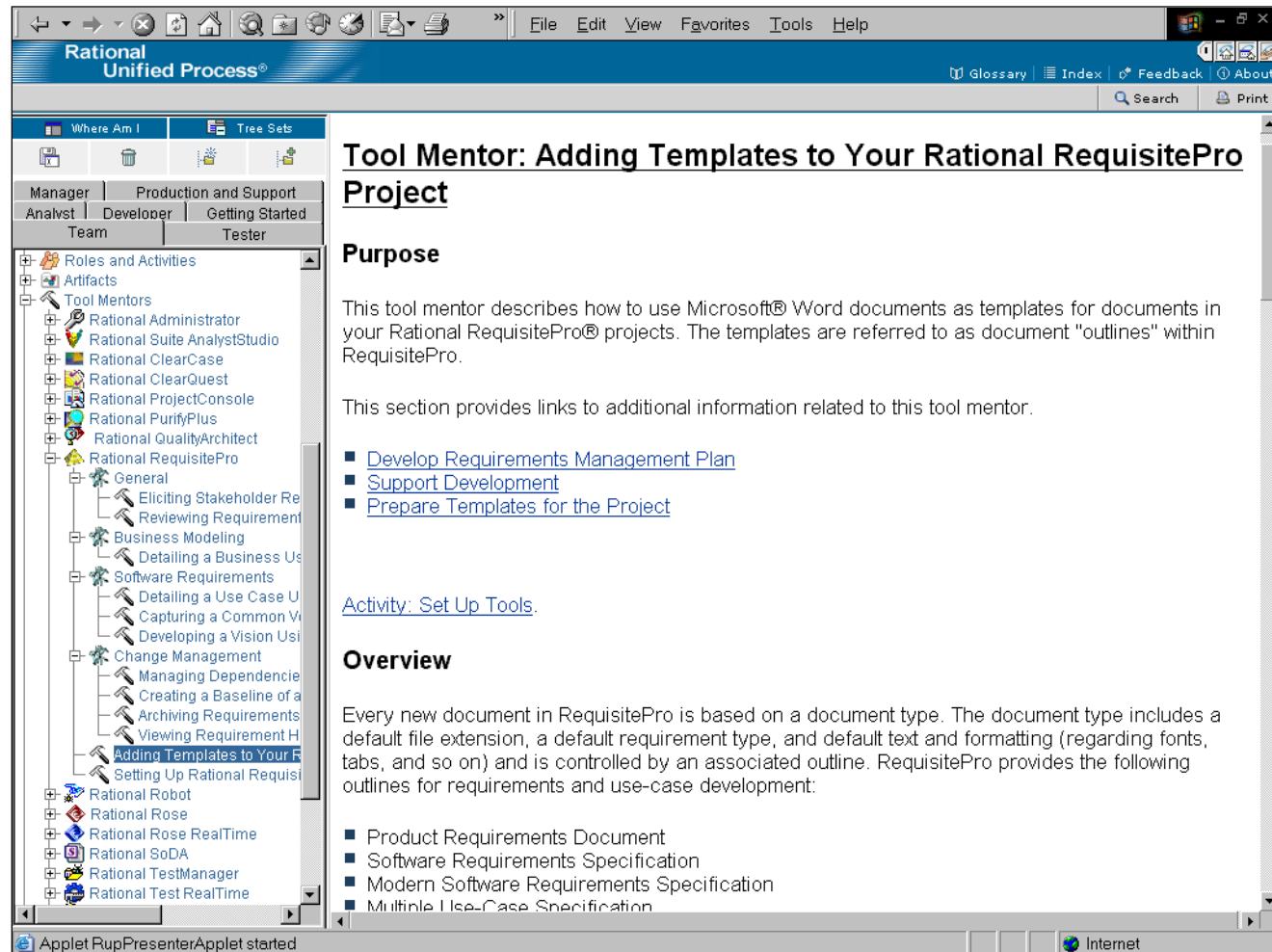
The bottom of the window shows a status bar with 'Applet RupPresenterApplet started' and an 'Internet' icon.

Project Management Templates



- Summary MS Project Example
- Detail MS Project Example

Tool Mentors - How to Use Rational Tools in RUP™



The screenshot displays the Rational Unified Process web application interface. The left sidebar shows a tree structure with 'Tool Mentors' expanded, listing various Rational tools. The main content area is titled 'Tool Mentor: Adding Templates to Your Rational RequisitePro Project'. It includes a 'Purpose' section with a paragraph about using Microsoft Word documents as templates, a list of links for additional information, an 'Activity: Set Up Tools' link, an 'Overview' section with a paragraph about document types, and a list of document types provided by RequisitePro.

Tool Mentor: Adding Templates to Your Rational RequisitePro Project

Purpose

This tool mentor describes how to use Microsoft® Word documents as templates for documents in your Rational RequisitePro® projects. The templates are referred to as document "outlines" within RequisitePro.

This section provides links to additional information related to this tool mentor.

- [Develop Requirements Management Plan](#)
- [Support Development](#)
- [Prepare Templates for the Project](#)

[Activity: Set Up Tools.](#)

Overview

Every new document in RequisitePro is based on a document type. The document type includes a default file extension, a default requirement type, and default text and formatting (regarding fonts, tabs, and so on) and is controlled by an associated outline. RequisitePro provides the following outlines for requirements and use-case development:

- Product Requirements Document
- Software Requirements Specification
- Modern Software Requirements Specification
- Multiple Use-Case Specification



Caveats

- RUP™ is far from complete
 - Focused on software development
 - Series of books on *Unified Process ... Phase* by Scott Ambler and Larry Constantine provide “missing” coverage
- Project oriented
- Lack of comprehensive “book” makes learning difficult



More Caveats

- IBM influence is creating some confusion and inconsistencies
- Use Cases are insufficient for good requirements, IMHO
- Fails to adequately address “data intensive” applications
 - Only addresses database design, no place for “data requirements”
- Viewed as “silver bullet” by many



Summary

- Forms solid basis for improving software development process, particularly for ad-hoc, Level 1 organizations
- Provides basis for incorporating OOA/OOD/UML into current software development process
- Provides basis for development using IBM, Rational and Microsoft technologies
- 30 day on-line evaluation available, <http://www.rational.com>



Questions
