Rational Unified Process
Process Description and Workflows

Mike Fourman
The RUP uses four elements to describe processes:

- **Workers** - describe a role, some people have many roles.
- **Activities** - small, definable, reusable tasks that can be allocated to a single worker.
- **Artifacts** - usually process deliverables, like: use cases, code, plans, test cases, test results.
- **Workflows** - coordinated sequences of activities.
• There are 9 workflows, 6 engineering workflows:
  – Business modelling
  – Requirements
  – Development & Analysis
  – Implementation
  – Test
  – Deployment
• And 3 supporting workflows:
  – Project management
  – Configuration and Change Management
  – Environment
Requirements Workflow

- Workflows are followed iteratively through each iteration of each phase of the RUP.
- Effort expended in each workflow depends on the phase.
- We describe the requirements workflow in more detail
Capturing/discovering requirements

- More difficult and more important than writing code
- Users know what they have, not what they need
  - They will better understand what they need after they see it.
  - Nobody needed the WWW until they saw it.
  - User cannot envision the possibilities enabled by technology
Capturing/discovering requirements (continued)

- Developer is rarely the user
- Diversity of users
- Support the user’s mission, not only the user
- User needs and missions are constantly changing
- The new system will impact the user’s needs, resulting in new system needs (cycle)
- Complex systems are never fully understood
  - understanding evolves as the system evolves
- Hard to understand the constraints of legacy systems and the system environment
Purpose of requirements workflow

• Aim development toward the right system (I.e. support validation)
  – Other workflows focus on building the system right (I.e. support verification.)
• Describe what the system should and should not do
  – an agreement between customer (including user) and development organization
  – in the language of the customer/user
Tasks in Requirements Workflow

- List candidate requirements
- Understand system context
- Capture functional requirements
- Capture non-functional requirements
- Validate requirements (not well-developed in RUP)
Starting points

Vague ↔ Overly Detailed

System Requirements

Vision statement

Detailed customer requirements document

Domain object model

Similar systems

Business model
List candidate requirements
→ Feature list

- **Candidate features that could become requirements**
  - Good ideas added to feature list
  - Features taken off list when they become formal requirements

- **Planning values**
  - Status
  - Cost
  - Priority
  - Risk
Understand system context

→ Business or domain model

• Domain model
  – Identify and name important concepts and entities in the system context
  – Identify and name relations between domain objects
  – Glossary for now, possible classes in analysis and design workflows
Business or domain model?

- Business model
  - Domain (object) model plus
    - processes/behaviors
    - workers, their responsibilities and operations
- Decide whether to build a business model, a domain model, or simply a glossary of terms
Capture functional requirements
→ Use cases

• Capture requirements as use cases
  - Use case: a user's way of using the system
  - When an actor (user or external subsystem) uses the system, the system performs a use case
  - All use cases = all the things the system must do

• Capture user interfaces that support the use cases
Capture non-functional requirements

→ Supplementary requirements
and use cases

• System properties
  – Environmental or implementation constraints
    ▪ e.g. must have remote access or must run on Linux or WinNT
  – Qualities (“-ilities”): performance, reliability, security, maintainability, extensibility, usability, etc.

• Tie to use cases or domain concepts, where possible
  – those that cannot be tied (they are general) are listed as supplementary requirements
Requirements in the life cycle phases

- **Inception**
  - identify most of the use cases to define scope
  - detail critical use cases (10%)
- **Elaboration**
  - detail the use cases (80% of the requirements)
- **Construction**
  - identify and detail remaining use cases
- **Transition**
  - track and capture requirements changes
Domain modeling

- Objects or concepts: things in the system context that the system must manipulate or keep track of
- Events that transpire in the system context
- Capture as class models or (for small systems) as a glossary of terms
- Creates a common language for customer and developer
- Focus on domain modeling; defer system internal modeling to analysis, design, and implementation
Business modeling

• Business use case model
  - processes (use cases) and users (actors) in roles
  - represents system from a usage perspective and outlines how it provides value to its users

• Business object model
  - how each use case is realized by a set of workers who are using business entities and work units
Requirements Workflow Description

- Artifacts created
- Workers participating
- Detailed workflow activity
Capturing Requirements as Use Cases

- Use cases
  - Offer a systematic and intuitive way to capture functional requirements
    - complement written documents of “the system shall ...” which are good for system test/verification
  - Focus on the value the system adds to each user or external system
  - Force analysts to think in terms of who the users are and their business or mission needs
  - Drive the other development workflows
Activity Overview

System Analyst:
- Develop Vision
- Capture a Common Vocabulary

Stakeholder:
- Elicit Stakeholder Requests

Dependencies:
- Manage Dependencies

Use Case Model:
- Structure the Use Case Model

Architect:
- Prioritize Use Cases

Use Case Specifier:
- Detail a Use Case

Software Requirements:
- Detail the Software Requirements

User Interface Designer:
- Model the User Interface

Prototype the User Interface

Requirements Reviewer:
- Review Requirements
A use case diagram identifies and names actors and their use cases.
Emergency Dispatch

Use Cases

Reports incident and associated details to dispatcher

Facilitates communication b/w System and other Actors:
- Receives incident report
- Logs details including: type and location
- Requests/recieves vehicle location info.
- Dispatches appropriate vehicle(s).
- Alerts other Emergency service(s).

AcceptIncidentReport

Obtain current location of vehicles

Make dispatch decision based off:
- type of incident
- location of incident
- location of available vehicles
Dispatch vehicle(s)

Alert Emergency Service

Alert other emergency service(s) if necessary.
Use-Case Model Artifact (Actors)

- **Actors**
  - Each type of user and each type of external system,
    - parties outside the system that interact with the system
    - Separate actor for each role of a user
  - There will be (at least) one system use case for each actor role
Use-Case Model Artifact
(Use Case, Structure)

- **Use Case**
  - Scenarios giving the ways actors use the system
    - Chunks of functionality the system offers to add a result of value to its actors
  - Specifies main sequence and alternative sequences of actions/events that the system can perform
  - Includes special requirements specific to the use-case

- **Can structure complex or large use case models**
  - perspectives, packages, etc.
**Other Requirements Workflow Artifacts**

- **Architectural view of use-case model**
  - shows the architecturally significant use cases
    - important and critical functionality
    - important requirements to develop early
    - input to use case prioritization
  - corresponding use case realizations usually appear in architectural view of analysis and design models

- **Glossary**
  - Important and common terms
  - Consensus agreement on meaning
  - Less formal view of business/domain model

- **User-interface prototype**
  - help understanding of user/system interaction
Worker Responsibilities

- **System analyst**
  - Identify actors and use cases
  - Create complete and consistent set of use cases and requirements (but not for details of each individual use case)
  - Develop glossary to facilitate complete/consistent requirements set

- **Use-case specifier**
  - Detail one or more use cases

- **User-interface designer**
  - Define the “visual shape” of the user interface for one or more actors
    - layout, behavior, inter-screen flow

- **Architect**
  - Describe architectural view of use-case model
Core Workflow

System Analyst

Find Actors and Use Cases

Prioritize Use Cases

Detail a Use Case

Structure the Use Case Model

Prototype User Interface

Use-Case Specifier

Repeat and Iterate

User-Interface Designer

2001-10-22 Cs2 Software Engineering
Activity: Find Actors and Use Cases

• **Purpose**
  - System boundary: delimit system from environment
  - Outline actors and their use cases
  - Capture and define common terms (glossary)

• **Inputs**
  - Stakeholders (especially customers, users, other analysts)
  - Business/domain model, vision document, customer requirements specification

• **Activity steps**
  - Find actors
  - Find use cases
  - Describe each use case
  - Describe use case model as a whole (including glossary)
Finding the Actors

- Example actors or actor categories
  - Business workers, business actors
  - The person/system asking the question, making the decision
  - External systems
  - System maintenance and operational support

- Criteria
  - Must be at least one real user who can enact the candidate actor
  - Minimum overlap between roles

- Capture
  - Actor name
  - What the actor uses the system for; actor (role) needs and system responsibilities
  - What the system uses the actor for; actor (role) responsibilities and system needs
Finding the Use Cases

- **Examples:** deliver an observable *result of value* to a *particular actor*
  - Use case(s) for every role of every worker
  - Use case to support user’s need to create, change, track, remove or study business objects
  - Use case to allow user to tell system of event or for system to tell user of event
  - Use cases for system startup, termination or maintenance
- **Use case name:** verb phrase describing result of interaction
- **Use case scope and boundaries** are hard to find
  - Decouple them in time and data sharing
  - Iterate with architecture tasks
Describing the Use Cases

• Description iterations add detail
  1. Name
  2. Few words
  3. Few sentences to summarize actions
  5. Formal models
Use Case Model as a Whole

- **Model as a whole**
  - *Glossary of common terms*
  - *Survey description: interaction of actors, interaction of use cases*
    - Use case generalizations and extensions
    - Flows between use cases
      - Activity diagrams
      - Post-conditions of one use case establish pre-conditions of others
  - *Group by business use case, by actor, by coupling, etc.*
Review

• Review items
  - Use case list is complete
  - Sequences of actions are correct, complete and understandable
  - All use cases have value to actors
Prioritize and Detail Use Cases

• Prioritize based on clarity of system scope, architecture impact, and risk
• Detail event flow, use case start/end, actor interaction
  – Pre-conditions; basic flow through states; post-conditions
  – Alternative flows to accommodate...
    ▪ actor choice
    ▪ influence of other actors
    ▪ actor error
    ▪ system error or failure
Detailed Use Case Description

- Start states (preconditions)
- The first action to perform
- Action order
  - Basic/normal
  - Alternates
  - Constraints
- How the use case ends
- Possible end states (postconditions)

- System interaction with actor and what they exchange
  - Clarify which does what
    - Actor initiation, system response
    - System initiation, actor response
  - If “actor” is an external system, specify protocol
- Usage of system objects, values and resources
- Non-functional requirements

**Review:**
understandable, correct, complete, consistent
**Activity: Prototype User Interfaces**

- **Inputs to activity:** use-case model, detailed use-case descriptions, supplementary requirements, glossary (or business/domain model)
- **Actor interacts by viewing and manipulating elements that represent attributes of use cases**
  - Assure each use case is accessible to the actors through the user interface
  - Assure well-integrated, easy-to-use, consistent, navigable user interfaces
    - Analyze usability; don't be fooled by wording of use case
- **Build physical prototype to validate UI and the use cases**
Activity: Structure the Use-Case Model

- General and shared functionality: “uses”
  - Like inheritance: specific (real) uses general (abstract)
  - The generalization captures overlap between use cases
- Additional or optional functionality: “extends”
- Be careful in structuring use-case model
  - Reflect real use cases
  - Keep things understandable and manageable
  - Decompose functionality in the analysis model, not the use-case model
    - Object-oriented decomposition, not functional decomposition
Summary of Requirements Workflow

• Capture requirements as
  - Business model, domain model or glossary to capture system context
  - Use-case model that captures functional requirements and use-case-specific nonfunctional requirements
    ▪ Survey description of model as a whole
    ▪ Set of diagrams
    ▪ Detailed description of each use case
  - Set of user-interface sketches/prototypes for each actor
  - Supplementary requirements specification for requirements not specific to a use case

• Next steps
  - Use cases drive use-case realization in analysis and design and test cases in testing
  - Analysis: reformulate use cases as interacting objects