Chapter 5
Requirements Elicitation

Object-Oriented Software Construction

Armin B. Cremers, Dr. Sascha Alda & Tobias Rho
(based on Bruegge & Dutoit)
Overview

- Introduction to requirements engineering
- General view on requirements elicitation
- Process of requirements elicitation (and analysis)
- Elicitation Techniques
  - Scenarios
  - Interviews
  - Observation
- From scenarios to use cases
- Conclusions
Software Development Process: A Brief Overview

Use Case Model

Requirements Elicitation

Analysis
System Design
Object Design
Implementation
Testing

Expressed in Terms of
Structured by
Realized by
Implemented by
Verified by

Application Domain Objects
Sub-systems
Solution Domain Objects
Source Code
Test Cases

Armin B. Cremers, Sascha Alda & Tobias Rho (based on Bruegge & Dutoit)
Object-Oriented Software Construction
First View on Requirements Engineering

- Requirements Engineering is the first phase of the Software Lifecycle
- Production of a specification from informal ideas
- Goal: Requirements Specification
  - System requirements specification: requirements describe Software and Hardware
  - Software requirements specification: describe only Software
- RE is about *what* the system should do (not *how* to do it)
- Key influencing factor to the development process
  - Failures made here result in high costs in later development phases
  - System will meet the user/customer needs
Requirements Engineering: Input and Output

- Initial Input
  - A Vision of a system to be created (vague)
  - Different stakeholders with different interests
  - → Problem Statement

- Desired Output
  - Specification as complete as possible
    - Complete coverage of the problem (all relevant requirements are captured)
    - Complete and exact definition of each requirement
Requirements Elicitation
First view

- Encompass all activities involved in discovering the requirements of a system
- System developers and engineers work in close relationship with customer and end-users to
  - Find out more about the problem to be solved
  - To describe the functionality of the system
  - Understand the application domain ("speak its language")
  - Hardware constraints … and so forth
- Not just a simple process about fishing for requirements, but a highly complex process:
  - Customer rarely have a clear picture of their requirements
  - Different people have conflicting requirements
Requirements Elicitation
Relation to Requirements Analysis

- Basis for Discussions with customer
- Definition of the system in terms understood by the customer ("Problem Description")

+ Design basis for developers
+ Technical specification of the system in terms understood by the developer ("Problem Specification")
**Process of Requirements Elicitation:**

**Products of Requirements Process**

- **Problem Statement**
- **Requirements specification:**
  - functional and non-functional requirements
- **Analysis Model:**
  - dynamic model
  - object model

**Requirements Elicitation**

**Requirements Analysis**
Requirements and their Meaning

- Definition of term “Requirement”
  - A condition or capability needed by a user to solve a problem or to achieve an objective
  - Condition or capability that must be met by a system
    - Satisfies a contract, standard, specification
  - Recall from the first lesson:
    - Requirements = Information, Ideas, Constraints
Functional and Non-Functional Requirements

- **Functional requirements**
  - Describe the interactions between the system and its environment independent from implementation

- **Non-functional requirements (Most typical)**
  - Quality aspects of the system not directly related to functional behavior.
  - Reliability, Performance, Availability, Supportability, Usability, Tailorability, Security

- **Pseudo requirements (Non-functional requirements B)**
  - Imposed by the client or the environment in which the system operates
  - Legal requirements
  - Design and Implementation Constraints

- **Project Management (Non-functional requirements C)**
  - Budget, Release Date
The Goal: Analysis Model (vs. Requirements Specification)

- Both models focus on the requirements from the user’s view of the system.
- Requirements specification uses natural language (derived from the problem statement).
- The analysis model uses formal (Z, pi-calculus) or semi-formal notation (for example, a graphical language like UML).
  - Formal notations encompass an exact mathematical syntax and semantic.
- The starting point is the problem statement.
Starting with the Problem Statement

- The problem statement is developed by the client as a condensed description of the requirements that should be addressed by the system.
- Describes the problem that should be solved.
- It describes “what” is needed, not “how” it should be reached.
Starting with the Problem Statement: Ingredients

- Current situation: The Problem to be solved
  - A few pages
- Description of one or more scenarios
- Some initial requirements
  - Functional and Non-functional requirements
  - No complete description
- Project Schedule
  - Major milestones that involve interaction with the client including deadline for delivery of the system
- Target environment
  - The environment in which the delivered system has to perform a specified set of system tests
- Client Acceptance Criteria
  - Criteria for the system tests
Starting with the Problem Statement: Problem vs. Change

- There is a problem in the current situation
  - Examples:
    - The response time in a travel booking system is far too slow
    - There have been illegal attacks to the system

- A change either in the application domain or in the solution domain has appeared
  - Change in the application domain
    - A new function (business process) is introduced into the business
    - Example: A function is provided for credit payment with fingerprint as authorization
  - Change in the solution domain
    - A new solution (technology enabler) has appeared
    - Example: New standards (implementation) for secure network communication
Example: Library System

- Idea: A Library Management System should be designed. Information on books, CDs, DVDs, Journals, etc. can be stored and retrieved

- Possible Requirements:
  - Searching by Title, Author, and/or ISDN should be possible
  - User Interface should be web-based (accessible via WWW Browser)
  - At least 20 transactions per seconds should be possible
  - All services should be available within 10 minutes
  - Users have no access to personal data of other users

Problem Statement

- functional requirement
- Implementation requirement
- performance requirement
- availability requirement
- Security requirement
Process of Requirements Elicitation: Activities during Requirements Elicitation

- Identifying Actors
  - Types of users, roles, external systems

- Identifying Scenarios
  - Interactions between users and the systems (one possible case)
  - Later on in this lesson

- Identifying Use Cases
  - Abstractions of Scenarios
    (Many possible cases)

- Refining Use Cases
  - Refinements, adding exceptions, etc.

- Identifying Relationships among Use Cases

- Identifying Non-Functional Requirements
  - Security issues, Performance, etc.
Process of Requirements Elicitation: How to elicit Requirements?

- **Sources of information**
  - Documents about the application domain
  - Manual and technical documents of legacy systems

- **User Participation**
  - Elicitation Techniques (see next slides)

- **Approach**
  - First describe a set of scenarios with elicitation techniques
  - Then aggregate the identified scenarios towards use cases
Elicitation techniques - Idea

- Specific techniques which may be used to collect knowledge about system requirements

- Requirements elicitation is cooperative process involving requirements engineers and system stakeholders. Problems:
  - Not enough time for elicitation
  - Inadequate preparation by engineers
  - Stakeholders are unconvinced of the need for a new system

- Types of Selection Criteria:
  - Interviews
  - Observations
  - Scenarios
  - Brainstorming
Selection Criteria

- System to be created (I)
  - Greenfield Engineering (completely new)
  - Reengineering (revise an existing system)
  - Interface Engineering (put a new front to an existing system)

- System to be created (II)
  - Highly interactive (Cooperation Support System)
  - Specific applications like Games

- Budget/Time

- Degree of User Participation
  - Time
  - Experience of users

- … (many more)
Interviews
Essentials

- Probably the most common technique of requirements elicitation.
- Interviewers must be open-minded and should not approach the interview with pre-conceived notions about what is required.
- Stakeholders must be given a starting point for discussion:
  - a question
  - a requirements proposal
  - an existing system
- Interviewers must be aware of organizational politics:
  - Some requirements may not be discussed because of their political implications.
- Types of interviews:
  - Structured vs. unstructured
  - Oral vs. written interviews
  - Interview of a group vs. a single person
Interviews: Different Goals

- During elicitation (early)
  - Understanding role of interviewee within organization
  - Understanding the work context
  - Getting requirements on new system

  **Goal: Description of complete scenarios**

- During analysis
  - Discussing use cases with client and users
  - Correction and refinement (requirements and functionality)

  **Goal: Getting complete use cases**
People often find it hard to describe what they do because it is so natural to them.

Actual work processes often differ from formal, prescribed processes

→ Sometimes the best way to understand it is to observe them at work

Approach: adopt methods e.g. from the social sciences which has proved to be valuable in understanding actual work processes

Suitable Approach: Ethnography (Lecture ORE)
Scenarios - Overview 1

Motivation (Observation):

- System stakeholders find it more intuitive to reason about concrete examples rather than abstract descriptions of the functions provided by a system (use cases).

Solution: Scenario

- “A narrative description of what people do and experience as they try to make use of computer systems and applications” [M. Carrol, Scenario-based Design, Wiley, 1995]
- A concrete, focused, informal description of a single feature of the system used by a single actor.
- Discovering scenarios exposes possible system interactions and reveals system facilities which may be required.
Scenarios - Overview 2

- Scenarios are *stories* which explain how a system might be used. They should include:
  - a description of the system state before entering the scenario
  - the normal flow of events in the scenario
  - exceptions to the normal flow of events
  - information about concurrent activities
  - a description of the system state at the end of the scenario

- Scenarios can have many different uses during the software lifecycle:
  - **Requirements Elicitation**: As-is scenario, visionary scenario
  - **Client Acceptance Test**: Evaluation scenario
  - **System Deployment**: Training scenario.
Scenarios: Different Types

- As-is scenario
  - Used in describing a current situation
  - Usually used in re-engineering projects
  - The user describes the system

- Visionary scenario
  - Used to describe a future system
  - Usually used in Greenfield engineering and reengineering projects
  - Can often not be done by the user or developer alone
    - brainstorming sessions

- Evaluation scenario
  - User tasks against which the system is to be evaluated

- Training scenario
  - Step by step instructions that guide a novice user through a system
Process of Requirements Elicitation: The Requirements Elicitation Cycle

- Observing users
- Interviewing users and clients
- As-Is Scenarios
- Visionary Scenarios
- Use Cases + Refinements
- Prototypes
- Stability Requirements Specification (System Specification)
  - Functional Requirements
  - Non-Functional Requirements
  - Use Cases
  - Scenarios
Scenarios:
Example - Accident Management System

Your Task (Problem Statement):
- Build a requirements model for a system that allows to report fire incidents. It should be able to report incidents for many types of buildings and things.

The approach: Start with single Scenario, e.g. “Warehouse in fire”. Interview Guideline:
- What do you need to do if a person reports “Warehouse on Fire?”
- Who is involved in reporting an incident?
- What does the system do, if no fire cars are available?
- Can the system cope with a simultaneous incident report “Warehouse on Fire?”
- What do you need to do if the “Warehouse on Fire” turns into a “Cat in the Tree”?
Scenario:
Example - Warehouse on Fire (Bruegge)

- **Bob**, driving down main street in his patrol car notices smoke coming out of a warehouse. His partner, **Alice**, reports the emergency from her car by using the SYSTEM.

- **Alice** enters the address of the building, a brief description of its location (i.e., north west corner), and an emergency level. In addition to a fire unit, she requests several paramedic units on the scene. She confirms her input and waits for an acknowledgment.

- **John**, the Dispatcher, is alerted to the emergency by a beep of his workstation. He reviews the information submitted by Alice and acknowledges the report. He allocates a fire unit and two paramedic units to the Incident site and sends their estimated arrival time (ETA) to Alice.

- **Alice** received the acknowledgment and the ETA.
Scenarios: Observations about “Warehouse on Fire”

- Concrete scenario
  - Describes a single instance of reporting a fire incident.
  - Does not describe all possible situations in which a fire can be reported.

- Normal behavior (“lucky day” scenario)
  - No exceptional cases

- Participating actors
  - Bob, Alice and John = concrete names
Scenarios: Observations about “Warehouse on Fire”

- … ok, but we have even more scenarios available and identified:
  - Report fire in a car
  - Report flat on fire
  - Report cat on fire
  - Report truck on fire

- Next step: aggregate these scenarios towards a coherent use case to describe the possible sequence of events to “report a fire incident”
Use Case: 
Example - ReportFireIncident

- The Fireman on duty notices a fire incident. The Fireman or his Replacement (hereafter termed Initiator) reports the emergency from their car by using the SYSTEM.

- The Initiator enters the address of the corresponding fireplace, a brief description of its location (i.e., north west corner), and an emergency level. In addition to a fire unit, the Initiator requests several paramedic units on the scene. He confirms his input and waits for an acknowledgment.

- The Dispatcher on duty, is alerted to the emergency by a beep of his workstation. He reviews the information submitted by the Initiator and acknowledges the report. He allocates a fire unit and a suitable number of paramedic units to the Incident site and sends their estimated arrival time back to the Initiator.

- The Initiator receives the acknowledgment and the ETA.
Use Case: Observations about “ReportFireIncident”

- A more abstract use case
  - Describes a potentially huge number of instances of reporting a fire incident,
  - Describe all possible situations in which a fire can be reported.

- Normal behavior ("lucky day" use case)
  - No exceptional cases

- Participating actors
  - Initiator, Fireman, Representative
From Scenarios to use cases
First pass

- Use case: an abstraction of possible coherent scenarios
- Scenario: a single example of a scenario
  → instance of a use case!

Example:

Use Case
"ReportFireIncident"

Scenario
"Report Warehouse on Fire"
Scenario
"Report Flat on Fire"
Scenario
"Report Car on Fire"
From Scenarios to use cases
Relationship of events

- Use case: abstract events
- Scenario: concrete events

Scenario “Report Car on Fire”
- concrete event a
- concrete event b
  ....
- concrete event x

Use Case “ReportFireIncident”
- abstract event A
- abstract event B
  ....
- abstract event X

derived from

Example: “Bob enters data of the flat”
Example: “Fireman enters data of object”
How to create a use case from a set of scenarios? (1/7)

- Start with an arbitrary chosen scenario.
- Identify the actors taking part in it.
  - An actor is an abstraction of (or role assumed by) concrete persons, a subject or entities.
    Example: “Bob” can be seen as an instance of an actor named “Fireman“
  - Identify the actor that initiates the use case ("primary" actor)
    → inspect substantives!
  - Identify the "secondary" actors, who typically react to the system rather than taking initiative themselves.
- Create a new Use Case bubble and Symbols for all involved Actors. Connect each of the actors with the use case.
  - For primary actors: annotate them with <<initiates>>
How to create a use case from a set of scenarios? (2/7)

- Write down the flow of events of the use case. For the first scenario under inspection, this is mostly a copy & paste operation:
- Take the events of the scenarios, replace references to concrete concepts with abstractions:
  - Person names (e.g. “Bob” → “User”)
  - Attributes (e.g. skip “on the road to home”)
  - Locations (e.g. “Flat” → “Fireplace”)
  - Job specifications (e.g. “Enter data with a Palm PDA OS 4.0” → “Enter data with a user terminal”)
As long as there are scenarios remaining, repeat the following:

- Pick a scenario that is not dealt with yet.
- If the scenario is *exactly* an instance of one of the use cases in your current model, you can just skip it.

If there is *no* matching, then create a new use case.
Update the use case if there are some incompatible events

- Some concrete Event cannot be represented by an abstract event
- The number of concrete events does not fit the number of abstract events

What to do? → Exercise 6!
How to create a use case from a set of scenarios? (5/7)

- If you detect (partial) scenarios that can be potentially shared by many use case, include them (reuse):

```
Scenario
"aScenario"
• concrete event a
• concrete event x

Use Case
"Supplier"
• abstract event A
• abstract event X

Use Case
"aBaseUseCase"
• abstract event K
• INCLUDE Supplier
• abstract event L
```

Diagram:

- Use Case "Supplier" <<include>> Use Case "aBaseUseCase"
- Use Case "anotherBaseUseCase" <<include>> Use Case "Supplier"
- Use Case "Supplier" <<include>> Use Case "anotherBaseUseCase"
How to create a use case from a set of scenarios? (6/7)

- If you think the new scenario represents optional or exceptional behavior, introduce an extension point in the flow of the original use case, and add the diverging behavior as an extension:

  Scenario “aScenario”
  • concrete event a
  • SKIP remaining events in case of an exception

  Use Case “Supplier”
  • abstract event A
  • SKIP remaining events

  Use Case “aBaseUseCase”
  • abstract event K
  • EP “Exception”
  • abstract event L

  Use Case “anotherBaseUseCase”
  EP: Exception

  <<extend>> (Exception)

  Use Case “Supplier”

  Use Case “aBaseUseCase”
  EP: Exception

  Use Case “anotherBaseUseCase”
  EP: Exception

  Use Case “Supplier”

  Use Case “aBaseUseCase”
  EP: Exception

  Use Case “anotherBaseUseCase”
  EP: Exception
How to create a use case from a set of scenarios? (7/ 7)

- Some more Pseudo Codes can be used in textual use cases:
  - INCLUDE <use case name>
  - SKIP <events>
  - REPEAT n times (subsequence)
  - EP-Cross <extension point name> (denotes that this extension point is valid throughout the next events)
  - IF <condition> THEN <events> ELSE <events>
  - INHERIT <events>
  - OVERIDDE <event> <newEvent>

- Further Heuristics can be applied:
  - An actor only interacts with one use case (interface to system)
  - Number of use cases should moderate
  - More heuristics are discussed in ORE …
Scenarios:
Possible questions in an interview

- What are the primary tasks that the system needs to perform?
- How do you currently perform your primary task?
- Do you know about any kind of system or service that already fulfills some task?
- What data will the (main) actor create, store, change, remove or add in the system?
- Are there other actors in the system (explain the term actor!)
- Do the actors need assistance during carrying out their tasks?
- What external changes does the system need to know about?
- What changes or events will the actor of the system need to be informed about?
- What kind of exceptions can you suggest?
- Can actors interrupt a sequence of interaction? What happens, if so?
- What about extra-ordinary events and tasks?
The goal of this phase is a model representing the requirements of the system seen from the user’s perspective.

First steps are:
- Write the Problem Statement
- Elicit Requirements (with Interviews, task observation)

First step of elicitation is understanding scenarios

Requirements elicitation is a cyclic process