Specifying Requirements
with Use Case Diagrams
Outline

• Introduction
  • Use Case Diagrams
  • Writing Use Cases
  • Guidelines for Effective Use Cases
### Where are we?

<table>
<thead>
<tr>
<th>Phase</th>
<th>Actions</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiation</td>
<td>Raising a business need</td>
<td>Business documents</td>
</tr>
<tr>
<td>Requirements</td>
<td>Interviewing stakeholders, exploring the system environment</td>
<td>Organized documentation</td>
</tr>
<tr>
<td>Specification</td>
<td>Analyze the engineering aspect of the system, building system concepts</td>
<td>Formal specification</td>
</tr>
<tr>
<td>Design</td>
<td>Define architecture, components, data types, algorithms</td>
<td>Formal Specification</td>
</tr>
<tr>
<td>Implementation</td>
<td>Program, build, unit-testing, integrate, documentation</td>
<td>Testable system</td>
</tr>
<tr>
<td>Testing &amp; Integration</td>
<td>Integrate all components, verification, validation, installation, guidance</td>
<td>Testing results, Working sys</td>
</tr>
<tr>
<td>Maintenance</td>
<td>Bug fixes, modifications, adaptation</td>
<td>System versions</td>
</tr>
</tbody>
</table>
Source of Requirements

• Initial requirements come from the customer, by:
  – Documents, such as RFI/RFP
  – Meetings, reports

• Advanced requirements come from the analysts, after studying:
  – Scope and price
  – Feasibility (technological, organizational etc)
  – Prototypes

• Final requirements are stabilized in an iterative process.
Requirements vs. Design

• Requirements:
  – **What** the system should do
  – More abstract

• Design:
  – **How** the system should do it
  – More detailed
Types of Requirements

• Visible Functional Requirements
  – “The system will deliver cash to the customer”
  – “Cash will be delivered after card was taken out”

• Qualitative Requirements
  – “The authorization process will take no more than 1 sec”
  – “The user interface will be easy to use”

• Hidden Requirements
  – “Database maintenance processes will occur every night”
Use Cases

• A use case is a **contract** of an interaction between the system and an actor.
• A full use-case model comprise of:
  – A diagram, describing relations between use-cases and actors.
  – A document describing the use case in details.
Use Case Diagram Objective

1. Create a semi-formal model of the functional requirements
2. Analyze and define:
   - Scope
   - External interfaces
   - Scenarios and reactions
What Makes Good Use-Case Specification?

• Lack of ambiguity
  – Each requirement must be interpreted in a single manner.

• Completeness
  – The collection of all use cases is everything that can be done to/with the system.

• Consistency
  – Requirements should not conflict with each other. If there are, tradeoffs must be detected and discussed.

• Avoid design
  – Requirements should raise a need, not answer it.
Use Cases as Means of Communication

The use case should stimulate a discussion about **what** the system should do, mainly with people who are outside of the development team.
Outline

• Introduction

• Use Case **Diagrams**

• **Writing** Use Cases

• **Guidelines** for Effective Use Cases
A Simple Example

Handle Message

Handle Call

Bill Management

Cellular Phone

Customer

External Phone Company

System boundary

Association

Use Case

Actors

Introduction | Diagrams | Writing | Guidelines
Finding Actors

• External objects that produce/consume data:
  – Must serve as sources and destinations for data
  – Must be external to the system

Humans  Machines  External systems

Organizational Units  Sensors
Actors can be generalized

The child actor inherits all use-cases associations

Should be used if (and only if), the specific actor has more responsibility than the generalized one (i.e., associated with more use-cases)
Linking Use-Cases

• Linking enables flexibility in requirements specification
  – Isolating functionality
  – Enabling functionality sharing
  – Breaking functionality into manageable chunks

• Three mechanism are used:
  – Include
  – Extend
  – Inheritance
Use-Case Levels

Base Use Case: Used directly by the user

User goals

Sub-functionality

Perform Sale

Choose Products

Fill-in billing info

Alistair Cockburn “Writing Effective Use Cases”
The “Include” Construct

- Include is used when:
  - Decomposing complicated behavior
  - Centralizing common behavior
- The base use case explicitly incorporates the behavior of another use case at a location specified in the base.

Example:

```
Perform Sale <<include>> Fill-in billing info
```
The base use case can incorporate another use case at certain points, called extension points.

Note the direction of the arrow

- The base use-case does not know which use-case extends it
Example: Amazon

- **Product Page**
- **Shopping Cart**
- **Review Writing**
Example – cont’d

Customer

- Search Product
  - «include» View Product Details
  - «include» Navigate Deals
  - «include» Checkout
  - «include» Handle Order Status

- View Product Details
  - After page generation
  - «extend» Rank Supplier
  - «extend» Write Review
  - «extend» Add to cart
  - «extend» Login
  - «extend» user is not a member
  - «extend» Register

- Customer

Introduction | Diagrams | Writing | Guidelines
Generalization between Use-Cases

- The child use case inherits the behavior parent use case:
  - The interaction (described in the textual description)
  - Use case links (associations, include, extend, generalization)
- Child use-case can substitute parent Use case
- Overriding occurs through the textual description

---

Example

Handle Call

- 1. Transfer call to available representative
- 2. Mark representative as busy
- 3. Start record call
- 4. Stop record call
- 5. Log call details
- 6. Mark representative as free

Customer Representative

Handle Sale Call

Handle Technical Assistance Call

Tech Assistant Representative

---
Generalization Hazards

- Combining generalizations of actors and use-cases can be dangerous

**Bad:** Undergrad can submit thesis

**Good:** Only graduate student can submit thesis
Orange’s objective: Build a system that handles SMS messages, handles calls (for 2 and 3 generation phones), including conference calls and multiple calls from a single phone. The system must support users on the move.

Who are the actors?

The Cellular Phone  External Phone companies
Example: Cell Company System

- External Phone Company
  - Handle Cell Migration
    - <<include>>
  - Handle Call
    - <<include>>
    - while talking
    - <<extend>> {phone initiate call}
    - <<extend>> {incoming call}
  - Handle Voice Call
  - Handle Video Call
  - Handle Multiple Calls
  - Handle Video Call
  - 3G Phone
  - Cellular Phone
    - Handle SMS Message
      - <<include>>
Outline

• Introduction
• Use Case **Diagrams**
• **Writing** Use Cases
• **Guidelines** for Effective Use Cases


Structure of a Use Case Specification

X Name

X.1 Preconditions

X.2 Main Flow

X.3 Subflow

X.4 Alternatives flows

Alistair Cockburn
“Writing Effective Use Cases”
Preconditions

• What the system needs to be true before running the use-case.

• Examples
  – User account exists
  – User has enough money in her account
  – There is enough disk space
Main Scenario

- The success scenario is the main story-line of the use-case
- It is written under the assumption that everything is okay, no errors or problems occur, and it leads directly to the desired outcome of the use-case
- It is composed of a sequence of subflows
- Example:

1. Administrator enters course name, code and description
2. System validates course code
3. System adds the course to the db and shows a confirmation message
Guidelines for Effective Writing

• Use simple grammar
• Only one side (system or actor) is doing something in a single step
• Write from an “objective” point of view using active tense
  – Bad: “Get the amount form the user and give him the money”
• Any step should lead to some progress
  – Bad: “User click the enter key”
Subflow – cont’d

• Branches:
  – If the user has more than 10000$ in her account, the system presents a list of commercials
  – Otherwise…

• Repeats:
  1. User enters the name of the item he wishes to buy
  2. System presents the items
  3. User selects items to buy
  4. Systems adds the item to the shopping cart
  5. User repeats steps 1-4 until indicating he is done
Use-Cases – Common Mistakes

• Complex diagram
• No actor
• Too many user interface details
  – “User types ID and password, clicks OK or hits Enter”
• Very low goal details
  – User provides name
  – User provides address
  – User provides telephone number
  – …
Alternative Flows

• Used to describe exceptional functionality

• Examples:
  – Errors
  – Unusual or rare cases
  – Failures
  – Starting points
  – Endpoints
  – Shortcuts

Introduction | Diagrams | Writing | Guidelines
Alternative Flows - Example

• Errors:
  – “Case did not eject properly”
  – “Any network error occurred during steps 4-7”
  – “Any type of error occurred”

• Unusual or rare cases
  – “Credit card is defined as stolen”
  – “User selects to add a new word to the dictionary”

• Endpoints
  – “The system detects no more open issues”

• Shortcuts:
  – “The user can leave the use-case by clicking on the “esc” key
If a base use-case include another use-case, we will add a reference as a step:

1. System presents homepage
2. User performs login to the system
   OR
   <include: login to the system>
• Scenarios do not include direct references
• Instead, they include extension points, such as:
  
  User enters search string  
  System presents search results  
  Extension point: results presentations  
  OR  
  <extension point: results presentations>

• The extension use-case includes conditions in which the extension is being committed
  – Example: if the user belongs to the “rich clients” group
  – If more than two commercials were found
Outline

• Introduction
• Use Case Diagrams
• Writing Use Cases
• Guidelines for Effective Use Cases
How to Model?

**Bottom-up Process**

Starting with throwing all scenarios on the page, and then combining them:

- save
- print
- load
- Bullets format
- Paragraph format
- Font format
- Save as
- preview

**Top-down Process**

Starting with an overview of the system, and then splitting Use-cases:

- File actions
- Formattin g actions
- Viewing Actions
How to Model – cont’d

• Most of the analysis process are actually Combined
Combining Processes

• **Number Limit:**
  – The diagram should have between 3 to 10 **base** use-case. No more than 15 use cases (base + included + extending).

• **Abstraction:**
  – All use-cases should be in similar abstraction levels.

• **Size:**
  – Use cases should be described in half a page or more.

• **Interaction:**
  – Use-cases which are carried out as part of the same interaction.
Dividing Processes

• **Size:**
  – If a use-cases takes more than a page, consider include/extend

• **Weak dependency:**
  – If the dependency between two parts of a use-case is weak, they should be divided.
More Guidelines

• Factor out common usages that are required by multiple use cases
  – If the usage is required use <<include>>
  – If the base use case is complete and the usage may be optional, consider use <<extend>>

• A use case diagram should:
  – contain only use cases at the same level of abstraction
  – include only actors who are required
When Are we Done?

- When every actor is specified.
- When every functional requirement has a use-case which satisfies it.
- A tractability matrix can help us determine it:
Summary

✓ Introduction
  ▶ to the Unified Modeling Language (UML)
  ▶ To Use Case Diagram

✓ Use Case Diagrams
  ▶ Dual presentation of use-cases
  ▶ Include, Extend, Inheritance

✓ Writing Use Cases
  ▶ Preconditions & Post-conditions
  ▶ Main scenario vs. Alternative Flow

✓ Guidelines for Effective Use Cases