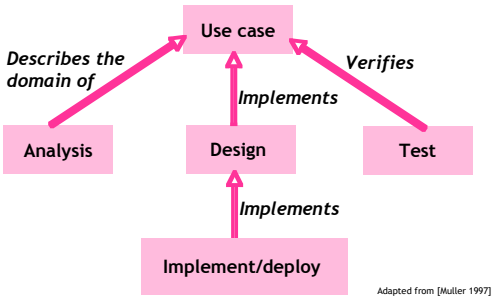
COMP2009
Software Engineering

Use Cases
Part I

USDP: Use Case Driven



Adapted from [Muller 1997] p160

Preliminary Activity - Domain Modelling

- The domain is the scope of the system being modelled.
  - The 'system' is the application *and* its environment.
- To construct use cases effectively we need to know the entities (classes) in the domain.
  - Things, objects.
  - E.g., Person, Book, Library, Catalog, etc.
  - Possible candidate classes.
- And their basic relationships.
- Use requirements and glossary to identify entities.

Example (see extended notes on Moodle)

- Identify named things (in red) as potential domain classes.

1. The **bookstore** will be web based initially, but it must have a sufficiently flexible architecture that alternative front-ends may be developed (Swing/applets, web services, etc.).
2. The bookstore must be able to sell **books**, with **orders** accepted over the **Internet**.
3. The user must be able to add books into an online **shopping cart**, prior to **checkout**.
   a. Similarly, the user must be able to remove **items** from the shopping cart.
4. The user must be able to maintain **wish lists** of books that he or she wants to purchase later.
5. The user must be able to cancel orders before they've shipped.
6. The user must be able to pay by **credit card** or **purchase order**.

Example (cont.)

- Review list to remove duplicates, actions, not in scope, etc.

Associate Partner	Customer Account	Order
Author	Customer Rating	Password
Book	Database	Purchase Order
Book Catalog	Editorial Review	Review Comment
Book Details	Internet	Search Method
Book List	Item	Search Results
Book Review	Keyword	Seller
Bookstore	List of Accounts	Shipping Fulfillment System
Category	Master Account List	Shopping Cart
Checkout	Master Book Catalog	Title
Credit Card	Master Catalog	User Account
Customer	Mini-Catalog	Wish List

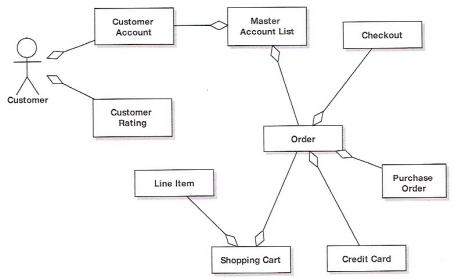
Example (cont.)

- Revised list

Associate Partner	Customer Account	Order
Author	Customer Rating	Purchase Order
Book	Database	Search Method
Book List	Editorial Review	Search Results
Book Review	Line Item	Shipping Fulfillment System
Category	Master Account List	Shopping Cart
Checkout	Master Book Catalog	Wish List
Credit Card	Mini-Catalog	

Example (cont.)

- First-pass domain model

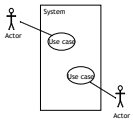


Example (cont.)

- Continue to refine.
- Look for generalisations.
- Don't spend excessive amount of time creating model.
- The entities/classes can then be used in the Use Case descriptions (see later).
  - Created a clear list of entities that are involved in system behaviours.
  - Avoid confusion and ambiguities.

Use Case Modelling Overview

- Use case modelling is a form of requirements engineering.
- Use case modelling proceeds as follows:
  - Find the system boundary
  - Find actors
  - Find use cases
  - Use case specification
  - Scenarios



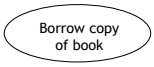
- It lets us identify the system boundary, who or what uses the system, and what functions the system should offer.

Use Cases v. Scenarios

- The full use case model consists of a collection of use cases.
- Each *use case* describes how a task (interaction) with the system is carried out.
  - As a dialogue between actor(s) and the system.
- A *scenario* is an instance of a use case as it would actually be carried out in a given set of circumstances.
  - One use case => many scenarios of the use case
- A use case *specification* must capture all possible scenarios including alternatives and errors.

Use Cases

- Use cases: Things actors do with the system.
  - A task that an actor needs to perform with the help of the system.
    - Example: Borrow a copy of a book.
  - A specific kind of system use; a 'case of use'.
- Represents the behaviour of a system from a user's standpoint by using actions and reactions.
  - This representation should be detailed design-independent.
- Use cases are triggered by an actor.

Use case icon: 

Identifying Use Cases

- Start with the candidate actors and consider.
  - What they need from the system.
    - What use cases there are that have value for them.
  - Any other interactions they expect to have with the system.
    - Which use cases they might take part in for someone else's benefit.
  - Distinguish between a 'normal' use and variants (alternatives and errors of the same use case).
  - Be prepared to add/remove/merge actors.
- Merge trivial use cases, split up long complex use cases.
- Describe system behaviour from an external point of view, avoid design, implementation or low level details.
- Iterate and review frequently!

Describing Use Cases

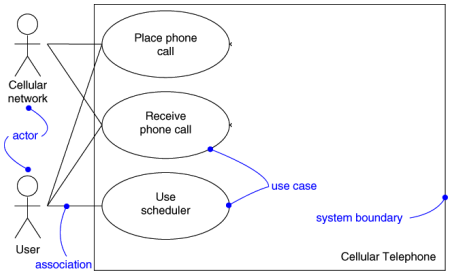
- Start with straightforward textual descriptions.
- Once the use case stabilises create a structured use case specification (coming up soon).
- The semantics are detailed in English text.
  - Third-person, active-voice.
  - Role playing is useful - see the CRC method notes.

Borrow copy of book
A BookBorrower presents a book to borrow. The System checks that the potential borrower is a member of the library, and that s/he has not already borrowed the maximum number of books. This maximum is six unless the member is a staff member, in which case it is twelve. If both checks succeed, the System records that this library member has borrowed this copy of the book. Otherwise, the System refuses the request.

Use the Domain Entities

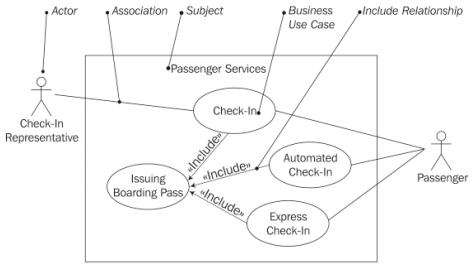
- "The user selects a title and adds it to the list of books saved for later. The system displays a page with the updated list and also shows a list of titles in the user's cart, ready for checkout."
- becomes:
- "The User selects a Book and adds it to their Wish List. The program displays a page with the updated list and also displays the user's Shopping Cart."
- Use of domain entities removes ambiguities and vagueness.
- Use the Actor names not other labels (e.g., 'The User selects' not 'a user selects').

UML Use Case Diagram Example (1)



Taken from [Booch 1999] RATIONAL SOFTWARE

The UML Use Case Diagram



UML2 in Action, Grässle et al.

Use Case Specification Template

There is no standard template defined by UML. A lot of variation in the literature.

Name: <Use case name/title>
ID: <Id code, e.g., UC5>
Brief Description: <One sentence overview>
Primary Actor(s): <Actor(s) that invoke the use case>
Secondary Actor(s): <Actor(s) that are invoked by the use case>
Preconditions: <Conditions that need to be true before use case starts>
Main Flow: <Interaction between system and actor(s)>
Postconditions: <Conditions that need to be true after use case ends>
Alternative Flows: <Alternative interactions to those in main flow>

Might also include:
Trigger: <What event triggers the use case to start>
Error flows: <Alternatives where the use case fails to complete at all>

Example Use Case Specification (1)

Use Case: EditProfile
ID: UC5
Brief descriptions: The applicant can edit their profile.
Primary actors: Applicant, Employer
Secondary actors: None
Preconditions: 1. The user must be logged onto the system
Main flow: 1. The use case starts when the user selects "Edit Profile" 2. The TAS displays a webpage showing the specific users account information 3. The user selects which field they want to edit. 4. Once this has been done the applicant user selects the "Save" tab. 5. The TAS will then return them to their homepage
Postconditions: 1. The TAS will update the users account information
Alternative flows: None.

Example Use Case Specification (2)

use case name	Use case: PaySalesTax
use case identifier	ID: 1
brief description	Brief description: Pay Sales Tax to the Tax Authority at the end of the business quarter.
the actors involved in the use case	Primary actors: Time
	Secondary actors: TaxAuthority
the system state before the use case can begin	Preconditions: 1. It is the end of the business quarter.
the actual steps of the use case	Main flow: implicit time actor 1. The use case starts when it is the end of the business quarter. 2. The system determines the amount of Sales Tax owed to the Tax Authority. 3. The system sends an electronic payment to the Tax Authority.
the system state when the use case has finished	Postconditions: 1. The Tax Authority receives the correct amount of Sales Tax.
alternative flows	Alternative flows: None.

Example Use Case Specification (3a)

USE CASE	CompleteFeedbackForm
ID	UC10
BRIEF DESCRIPTION	Voters are required to complete feedback form
PRIMARY ACTORS	Voters
SECONDARY ACTORS	None
PRECONDITIONS	Voters have already cast and confirmed their votes
MAIN FLOW	1)EVS displays feedback form 2)The voter fills in the form and submits 3)EVS store the details from the form 4) The system displays the elections screen
POST CONDITIONS	The voter has submitted form
ALTERNATIVE FLOWS	CancelForm

Example Use Case Specification (3b)

ALTERNATIVE FLOW	CompleteFeedbackForm:CancelForm
ID	UC10.1
BRIEF DESCRIPTION	The voter chooses to not to complete the feedback form
PRIMARY ACTORS	System
SECONDARY ACTORS	Voter
PRECONDITIONS	The voter has cast a vote
MAIN FLOW	1) The alternative flow starts after step 1 of the main flow 2) The voter selects the cancel option 3) The system displays the elections screen
POST CONDITIONS	None
ALTERNATIVE FLOWS	None

Diagrams v. Textual Specification

- The real value of use cases lies in creating the textual specifications.
  - Using a consistent template.
  - The template is not defined by UML.
- UML Use Case diagrams act as useful visual summary of actors, use cases and relationships.
  - But lack all the other important information in the textual version.
- Don't fall into trap of assuming diagrams are all there is.

Actors

- Actors: Who or what uses the system.
  - An actor is anything that interacts directly with the system.
    - Example: A person
    - Example: Another system
    - Example: A time-based trigger
  - An actor is a user of the system in a particular role.
- An actor is generally external to a system, though the system may hold an internal representation of the actor.
  - Example: BookBorrower
- An actor is depicted as a stick person.
- Actors trigger use cases.

Identifying Actors

- Observe the *direct* users of the system, which are those people and/or systems responsible for its installation, use or maintenance.
  - What roles do these users play in the interaction?
  - Who provides information to the system?
  - Who receives information from the system?
- The same physical person may play the role of several actors.
  - Example: A person may act as a Librarian and also as a BookBorrower
- Many people may play the same role and thus act as the same actor.
  - Example: Multiple people use a library as BookBorrowers.
- All of this becomes clearer as use cases are developed.

Describing Actors (Glossary Entry)

- Describe each actor clearly and precisely in a few lines of English.
  - Short name
  - Short description (semantics)

BookBorrower
This actor represents someone that makes use of the library for borrowing books

Further Guidelines

- Specifying a use case involves describing its semantics in English
- **Preconditions**: Clearly define the system state before the use case can begin.
  - Things that must be true before it begins.
- **Flow of events**: The steps in the use case corresponding to a normal scenario.
  - Keep it focussed and make sure all steps are included.
- **Postconditions**: Clearly define the system state after the use case has completed.
  - Things that must be true after it completes
- Can use structured English 'pseudo code' for more complex structure
  - **If**: To branch on different user actions
  - **For**: To repeat some actions
  - **While (condition)**: To repeat actions when something is true

If or loop v. Alternative flow

Main flow:
1. The use case starts when the actor clicks on "Remove Account" in the Administrator Interface.
2a. If the user searches for an account: <<include (SearchAccounts)>>
2b. If the user selects to displays all accounts, all the accounts are displayed
3. The actor selects the account to remove by clicking on the "Remove" button next to the desired user name (All accounts can be removed except the main "Administrator" account).
4. The system asks for a confirmation to delete the account.
5. The system displays whether the account was successfully removed.

- Use the approach that gives greatest clarity.
- In general:
  - If selection or repetition occurs within the use case but it still ends the same way, use if, while or for.
  - If the use case has multiple endings, use one or more alternatives.
- Also note the include in the example above, to avoid having duplicate sections of a flow.

Alternatives

- An alternative flow describes an alternative path through part of the use case.
- All alternatives must be captured to avoid a use case having dead end or missing paths.
- For example:
  - Main flow: Enter contact details -> successful
    - Alternative: Post code invalid -> rejected, display message to get user to enter valid post code.
  - Main flow: Checkout -> credit card validated, transaction complete.
    - Alternative: Credit card rejected -> request another card or let user cancel.

Requirements Traceability

- Use case models document requirements in a complementary way to traditional requirements documents.
- Items in these requirements documents should be linked to items in the use case models to ensure coverage, to help assess completeness of the requirements and to ensure consistency
  - Establishing this link enables requirements traceability.
  - This is essential for change management and is typically supported by tools.

use cases

requirements

	UC1	UC2	UC3
R1	x	x	
R2		x	
R3	x		x

Summary

- Use cases describe dialogues between actors and the system.
- A Use Case Diagram gives a visual summary of a set of Use Cases.
- Use Case Specifications textually describe a Use Case in detail.
- Use Cases must relate to requirements.
  - Traceability