

Use cases

Describing functional requirements

Introduction

- Use cases describe the functional requirements of a system
 - in a simple notation close to the final user,
 - yet formal enough to serve as a basis for a system design.
- They are elaborated during the requirements analysis phase
- But used over the entire product lifecycle
 - to plan and monitor progress
 - as a reference to ensure that the right system is being built
- Each use case describes a single functionality of a system from an external point of view
 - The system is treated as a black box
 - Internal implementation details are deliberately abstracted

Key concepts

- Use case
 - A functionality provided by the system being modeled.
 - GSM example : Place emergency call
- Actor
 - *Primary actor* : An external system or physical person which requires a given functionality from the system
 - *Secondary actor* : An external system or physical person which is used by the system during a use case.
 - GSM example : Caller (primary) Network (secondary)
- Scenario
 - A sequence of messages exchanged between the system and actor(s) during execution of a use case (*dialogue*).
 - GSM example : Caller requests 112, Phone emits "engaged" signal

Notation

- Use case description
 - Summarises all information relative to a single use case
 - Most importantly : Describes **all** possible scenarios (actor-system dialogues)
- Use case diagram
 - Summarises the actors, use cases, and the relationships between them
- The use case diagram is a summary of the use case descriptions;
 - the descriptions and diagram **MUST** be coherent.

Use case description

- *Name* (and often a number) : A short verbal phrase in the end-user vocabulary that indicates the function to be provided / the goal of the primary actor
- *Actor list* : The primary actor, and any secondary actors
- *Objective* : The objective of the primary actor, if not clear from the use case *name*
- *Nominal scenario* : The dialogue that takes place between the system and the actors under "normal" circumstances
- *Alternative scenarii* : Dialogues that takes place under "abnormal" circumstances, clearly specified

GSM Example : Description

Use Case 1: Place emergency call

Actor(s) : Caller (primary), Network (secondary)

Objective : Contact the emergency services

Nominal scenario 0

1. The caller enters the number 112 and presses "dial"
2. The GSM phone requests an emergency line from the Network
3. The Network connects to the emergency services and signals success.
4. The GSM phone transmits the bi-directional voice data

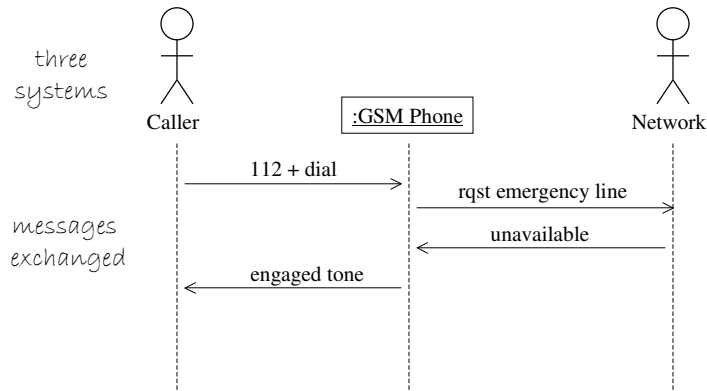
Alternative scenario 1: All emergency lines are occupied

After 0.2 :

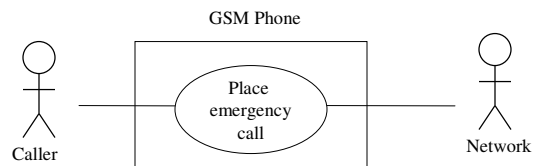
3. The network signals that all emergency lines are occupied.
4. The GSM phone emits an "engaged" signal.

System-level sequence diagram

- Represents a single use case scenario graphically



Use case diagram



- Summarises actors, use cases, and relationships between them
- An actor is represented by a stick figure labeled with its role; a use case by an oval labeled with its name.
- A solid line between an actor and a use case represents a relationship of communication, i.e. that this actor participates in this use case.
- Sometimes a solid rectangle is placed around the use cases to explicitly show the boundaries of the system.
- Primary actors are usually shown on the left, secondary actors on the right

Actors



- An actor is any physical person or external system which interacts with the system.
- More precisely, an actor is a **role** played by a physical person or an external system when interacting with the system.
 - A single role may be played by many different physical people (eg "Student")
 - A single person may play many roles (Mr Smith the "Bank Manager" and "Bank Client")
- While actors do not actually form a part of our system (they are external), identifying them helps us to identify **all** the functional requirements of the system (for all its users).

Identifying actors

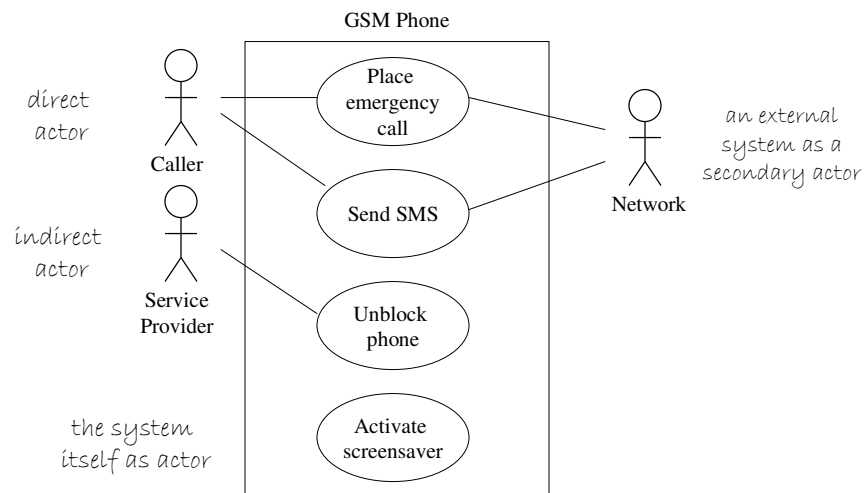


- Direct final users
 - The people who demanded the service in the first place
 - GSM example : Caller, Callee
- Indirect support users
 - People necessary to maintain the system
 - GSM example : repairman, service provider
- Other systems which communicate directly with the system
 - GSM example : The GSM network, a Bluetooth earpiece, another GSM phone
- The system itself
 - For "automatic" tasks or general capabilities
 - The actor may be called "System", "System Clock", etc
 - GSM example : activate screensaver, auto-shutdown, clock display

Identifying use cases

- Make a list of all primary actors, both direct and indirect human users and external systems.
 - *Who will use this system, and to do what ?*
- Determine what functions each of these actors requires from the system.
 - *What data will be stored by the system ?*
 - *What data does the system need to present to the users ?*
 - *What general capabilities must the system have ?*
- If the system uses the services of another system, add a secondary actor.
 - *What functions can we re-use or out-source ?*
 - *What existing or third-party systems do we need to integrate with ?*
- Each use case will need to be implemented, so carefully evaluate whether you really need it.

GSM Example



Elaborating use cases

- Create a list of actors and use cases for each actor.
- Describe the nominal scenario for each use case.
 - This gives you 70-80% of the system
- Optionally create a mock-up (maquette) of the use case interface
- Iterate, adding the alternative scenarios as your understanding of the system grows.
- Take care to abstract all internal system workings (black box) in the use cases.

Use case relationships

- An link between an actor and a use case indicates that the actor communicates with the system during that use case.
- We can also show several relationships between cases:
 - **include** allows us to factorise behaviour to avoid repetition
 - **generalisation** is used to casually describe an important variation on normal behaviour
 - **extend** is used to formally describe an important variation on normal behaviour by declaring "extension points"

include relationship

- Used when part of a scenario is shared by more than one use case, to avoid copy - paste.
- We factorise the shared behaviour and create a new use case related to the source use cases.

include relationship

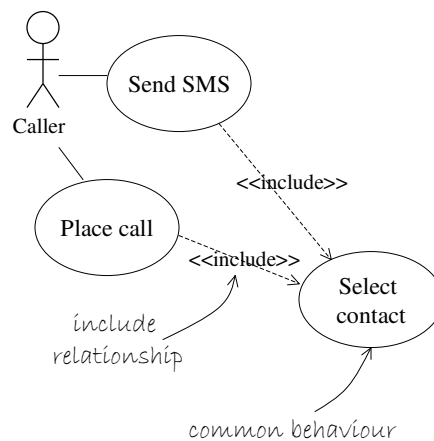
Cas 1: Send SMS

Actors : Caller, Network

Objective : Send a text message to another GSM user

Nominal scenario

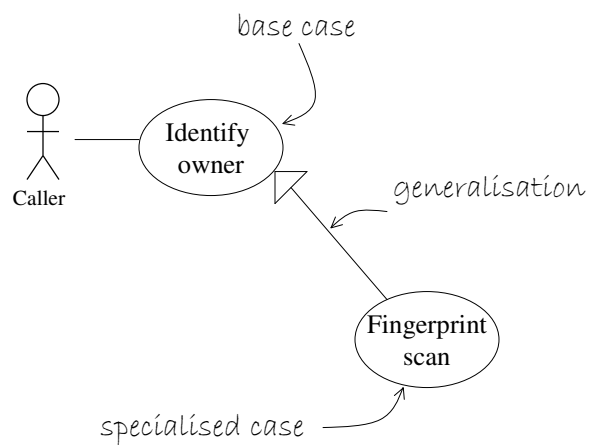
1. Caller selects "send SMS" function
2. Caller types in text message
3. include (Select contact)
4. Caller presses "send"
5. GSM phone asks Network to deliver the message (text, GSM number)
6. Network signals successful delivery.
7. GSM phone displays "SMS delivered".



Use case generalisation

- Used when one use case is similar to another, but does a little more.
- Its a way of exposing important alternative behaviours on the use case diagram.
 - The specialised case is just an alternative scenario of the base case
- Can also be used during system maintenance to add a new alternative behaviour without modifying existing behaviour.

Use case generalisation



extend relationship

- Similar to generalisation, but with more rules
- The specialised case adds behaviour to the base case, but the base case must declare "extension points" and the extending case can only add behaviour at these extension points.

extend relationship

Case 1 : Send SMS

Actors : Caller, Network

Objective : Send a text message to another GSM user

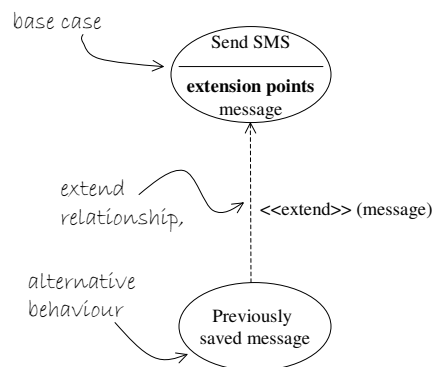
Nominal scenario

1. Caller selects "send SMS" function
2. Caller types in text message (extension point)
3. include (Select contact)
4. Caller presses "send"
- ...

Case 10 : Previously saved message

Nominal scenario

1. Caller selects SMS menu
2. Caller selects "Saved messages"
3. System displays a list of saved messages
4. Caller selects a message



Use case relationships

- Use case relationships add no meaning to the functional model, serving only as an organisational and documentation aid
 - avoiding cut and paste (include)
 - exposing important alternative scenarios (extend and generalisation)
 - splitting a complicated use case that is too large for a single construction iteration

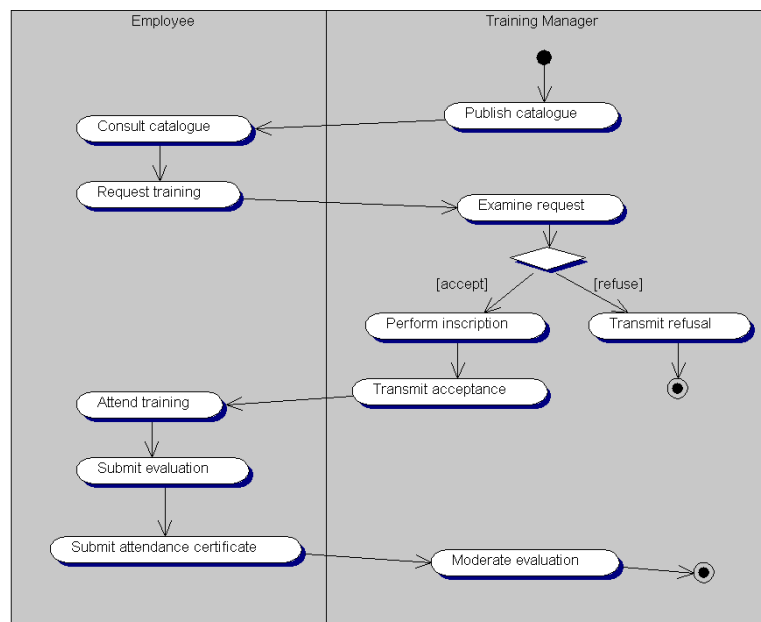
Example : Bank ATM

- The simple ATM's of the MoneyMoney Bank offer the following services :
 - Cash withdrawals to clients of the bank and holders of a *Carte Bleue*.
 - Consultation of bank balance and deposit of cash to clients of the bank.
- The ATM consists of
 - a standard user interface of screen and push-buttons,
 - a smart card reader,
 - automatic drawers for delivering and receiving money, which open and close automatically after 5s.
- The ATM always delivers cash in the smallest number of notes possible. It carries denominations of 10€, 20€ and 50€.
- Each ATM is associated to an agency, and when it runs low on cash, signals the manager of the agency, which replenishes it as necessary.
- Each ATM is regularly verified by a qualified MMB technician.

Business process modeling system use cases

- A **business process** is a sequence of activities performed by different actors during the operations of a business.
 - activities may be manual or computer-aided/automated
- **Business process modeling** is used before use case modeling to better understand a business and to identify potential use cases.
 - Described using activity diagrams in UML
- Use cases are then used to model the activities which can be automated by or supported by the information system.
 - Each use case models how the information system helps a single actor perform a single activity.
- The model can also be used to describe the ordering of (and dependencies between) use cases.

A business process model



Is my use case model ready?

- Have I produced a use case diagram and a complete description for each use case shown thereon?
- Are these documents coherent?
- Have I respected the client's vocabulary?
- For each use case description:
 - Is the name a short verbal phrase giving a clear and precise idea of its function?
 - Is the objective well-defined?
 - Have I recorded the primary actor? And all secondary actors?
 - Do my scenarios respect the "dialogue" format?
 - Can I think of any more alternative scenarios?
- Is all internal system detail abstracted?
- *Is the client satisfied with the use case scenarios (and/or maquettes)?*

Realising your use cases (1)

- Most methods based on UML build a system incrementally, use-case by use-case.
- We study the most important remaining use case
 - specify a complete architecture which supports this case,
 - construct a maquette and/or a prototype,
 - validate the prototype with the client ;
- And iterate (loop).
- Each new case contributes incrementally to the system.
- If a use case is too large for a single iteration, we can split it using the use case relationships (extend, generalisation)
- When all the cases have been realised, the system is complete.

Realising your use cases (2)

- The per-use-case architecture consists of :
- A **sequence diagram** for the nominal scenario and *interesting* alternative scenarios
 - A sequence diagram describes the collaboration of system objects to realise a use case behaviour.
- An increment of the **class diagram**
 - The class diagram describes the global data structure of the system.
- An increment to the **statechart(s)**
 - Statecharts describe the global behaviour of the system or a class.
- New **activity diagrams**
 - An activity diagram describes a detailed data manipulation algorithm used by the system.