

Use Cases

Use cases- what is it?

- Use cases are text stories, widely used to discover and record requirements; they are the **F** in FURPS+
 - A simple way to document functional requirements is to write use cases
- Use cases are not diagrams they are text : UML use case diagrams are trivial to learn; the real analysis skill is to identify and write good use cases.
- A use case describes how to achieve a specific goal, or function, using the system.
 - use cases will not hold all the requirements, but only describe the behavioral portion, the required functions of the system,
- A use case diagram is useful as an overview of the functionalities of an application; it also defines the application's context (what is part of it and what isn't).

Definitions

- **Actor:** something with behaviour, such as a person, computer system, or organization, *e.g. a customer, a player*.
 - An actor specifies a role played by a person, organizations or any other systems that interacts with the system.
- **Scenario:** specific sequence of actions and interactions between actors and the system. It is one particular story of using a system, or one path through a use case.
 - *e.g. the scenario of successfully purchasing items with cash.*
- **Use case:** a collection of related success and failure scenarios that describe actors using a system to support a goal.
- **A key point is to focus on the question:**
 - “How can using the system provide observable value to the user, or fulfill their goals?”

Use case example

- **Process Sale**
- A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the items.

Use case types and formats

- Uses cases may be written in three formality types
 - **Brief Use Case:** one-paragraph summary, usually of the main success scenario. Created during early requirements analysis, to get a quick sense of subject and scope. May take only a few minutes to create;
(e.g. *Process sale*)
 - **Casual Use Case:** Informal paragraph format. Multiple paragraphs that cover various scenarios. A refinement of a brief use case.
 - **Fully Dressed Use Case:** All steps and variations are written in detail, and there are supporting sections, such as preconditions and success guarantees.

Example of casual Use case

- **Handle return**
 - ***Main success scenario:*** A customer arrives at a checkout with items to return. The cashier uses the POS system to record each returned item...
- ***Alternate scenarios:***
 - If the customer paid by credit, and the reimbursement transaction to their credit card is rejected, inform the customer and pay them with cash.
 - If item identifier not found in the system, notify the Cashier and suggest manual entry of the identifier code.

Fully-dressed example: Process Sale

Use case UC1: Process Sale

Scope: POS application

Level: user goal

Primary Actor: Cashier

Stakeholders and Interests:

- Cashier: Wants accurate and fast entry, no payment errors, ...

- Salesperson: Wants sales commissions updated.

...

Preconditions: Cashier is identified and authenticated.

Success Guarantee (Postconditions):

- Sale is saved. Tax correctly calculated.

...

Main success scenario (or basic flow): [see next slide]

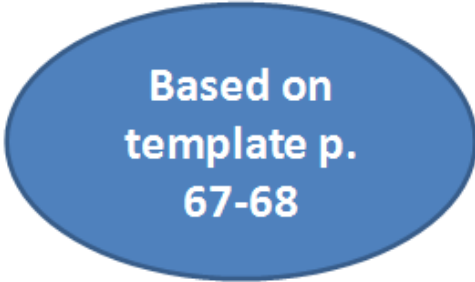
Extensions (or alternative flows): [see next slide]

Special requirements: Touch screen UI, ...

Technology and Data Variations List:

- Identifier entered by bar code scanner,...

Open issues: What are the tax law variations? ...



Based on
template p.
67-68

Fully dressed example: Process Sale (cont.)

Main success scenario (or basic flow):

The Customer arrives at a POS checkout with items to purchase.

The cashier records the identifier for each item. If there is more than one of the same item, the Cashier can enter the quantity as well.

The cashier
the r
item

On c
that

The System calculates and presents the sale total.

The Cashier tells the customer the total.

The Customer gives a cash payment ("cash tendered") possibly greater than the sale total.

**See full version with
numbering p. 68-72**

formation to

e of the current

OS system

Extensions (or alternative flows):

If invalid identifier entered. Indicate error.

If customer didn't have enough cash, cancel sales transaction.

Alternative notation

- A main alternative use case notation is to present the use case as a *conversation* between the actors and the system in a two column format:

Main Success Scenario:

Actor Action

1. Customer arrives at a POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale
3. Cashier enters item identifier.

Cashier repeats steps 3-4 until indicates done.

6. Cashier tells Customer the total, and asks for payment.
7. Customer pays.

System Responsibility

4. Records each sale line item, presents item description and running total.
5. Presents total with taxes calculated.
8. Handles payment.
- 9...

Three kinds of actors

- Actors are **roles** played by people, organisations, other software and devices that interact with the application.
- **Primary actors**
 - Those that have user goals fulfilled through using services of the system
- **Supporting actors**
 - Provides a service to the system
 - An actor needed to help carry out subgoals, such as a subsystem or another organization.
- **Offstage actor**
 - has an interest in the behaviour of the use case, but is not primary or supporting. (e.g. tax agency)

Goals and Scope of a Use Case

- At what level and scope should use cases be expressed?
 - Focus on uses cases at the level of **elementary business process (EBP)**.
 - **EBP: a task performed by one person in one place at one time which adds measurable business value and leaves the data in a consistent state.**

Goals and Scope of a Use Case

- **Are these valid use cases?**
 - Process Sale
 - Negotiate a supplier contract
 - Log in
 - Move Piece on Game Board
- A Use case consists normally of 5-10 steps. A simple small step is not a Use case
- It is usually useful to create separate “sub” use cases representing subtasks within a base use case.
 - e.g. Paying by credit

Goals and Scope of a Use Case

- **Are these valid use cases?**
- Process Sale **YES**
- Negotiate a supplier contract **NO**
- Log in **NO**
- Move Piece on Game Board **NO**

Guideline: Write Use Cases in UI-Free Style

- At this stage it is important to focus at the intent and keep the interface out. We are investigating the requirements; it is too early for design decisions.
 - e.g. Do not write:
 - “Admin enters ID and password in the dialog box
- A UI-free use case:
 - Is simpler and shorter;
 - Allows you to focus on the goals of the actors rather than how they will be achieved at a UI-level;
 - Does not preclude the development of UI prototype to help eliciting the requirements and record them in use cases;
- If a strict constraint exists (for whatever reason) it must be recorded in the **Supplementary Specification** document
- Also try to write short (but complete!)

Guideline: Write Black-Box Use Cases

- Consider the system as a black-box at this stage. The use cases must specify
- **what** the system must do **not how** it will do it
 - Do write '*the system records the sale*' rather than '*The system writes the sale to the database via ...*' or even worse '*the system generates an SQL INSERT statement for the sale*'
- We must consider the uses cases from the actors point of view only: to stress observable user value and focus on users' typical goals.
- We must concentrate at this early stage on what the user need not on how it will work internally.

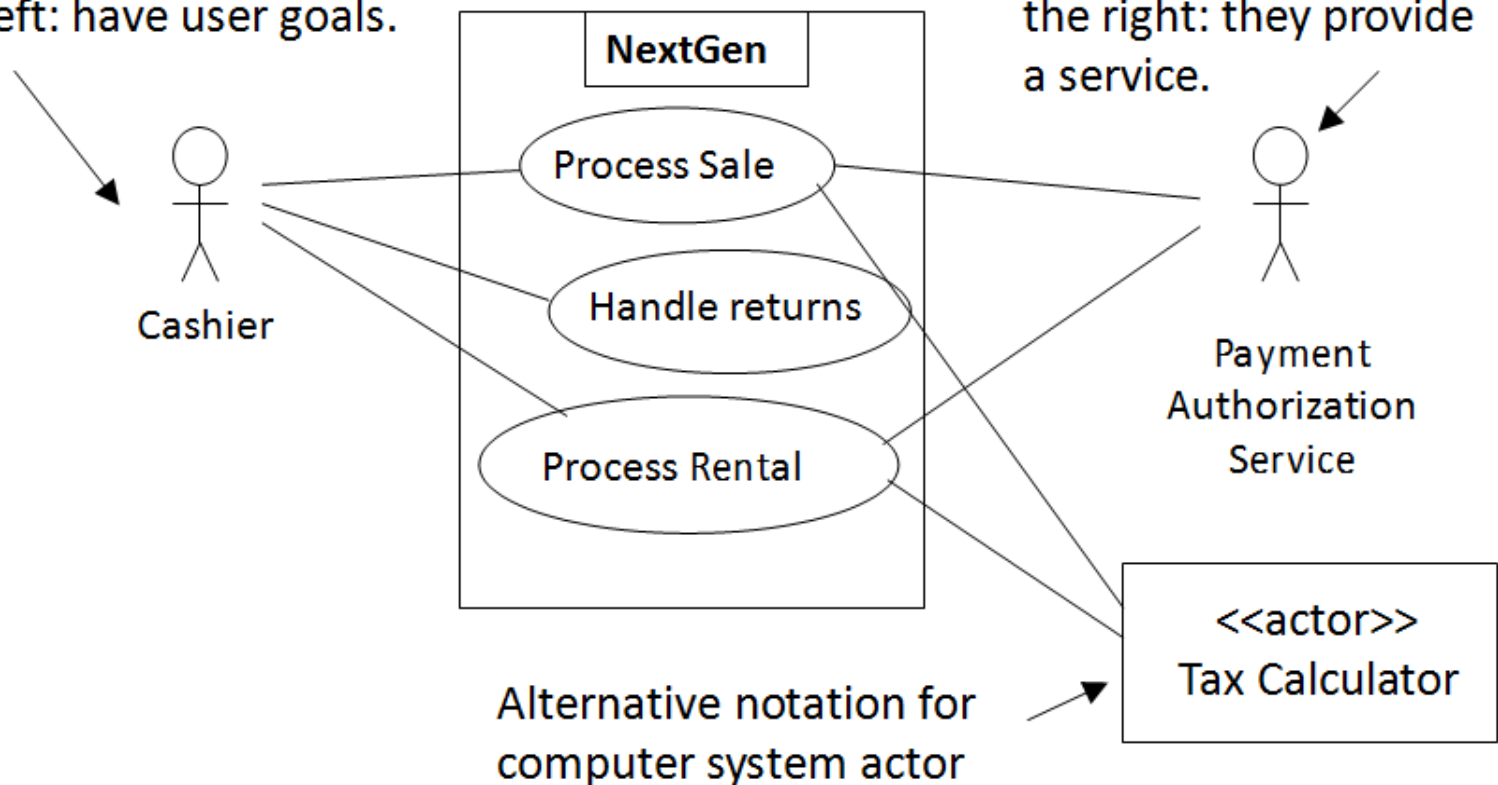
Finding primary actors, goals and use cases

- Choose the system boundary.
- Identify primary actors.
- For each actor identify their user goals.
- Define use cases that satisfy user goals; name them according to their goal. Name must start with a verb
- Exception to the rule one goal- one use case:
 - CRUD (create, read, update, delete) one Use case: ***Manage X***
- Of course, in iterative and evolutionary development, not all goals or use cases will be fully or correctly identified near the start. It's an evolving discovery.

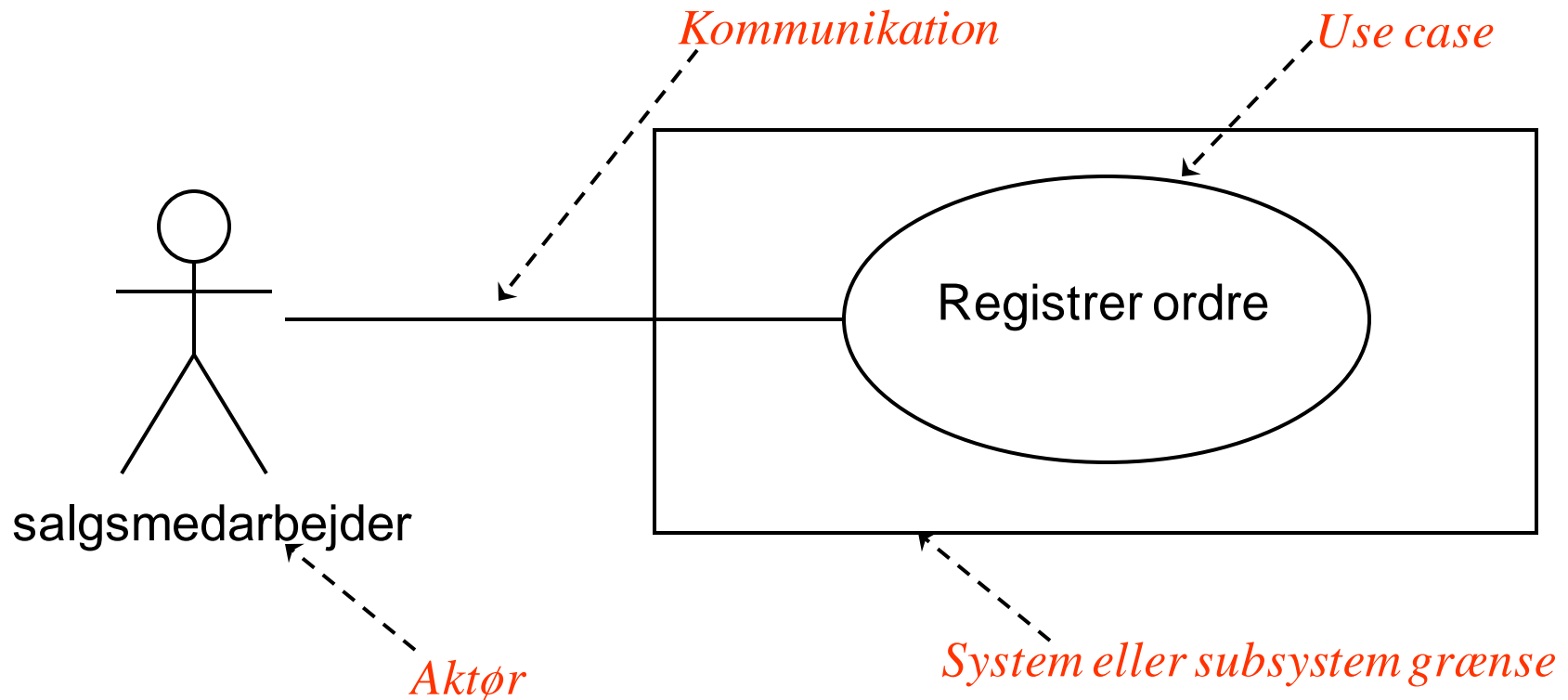
Use Case Diagrams

Primary actors to the left: have user goals.

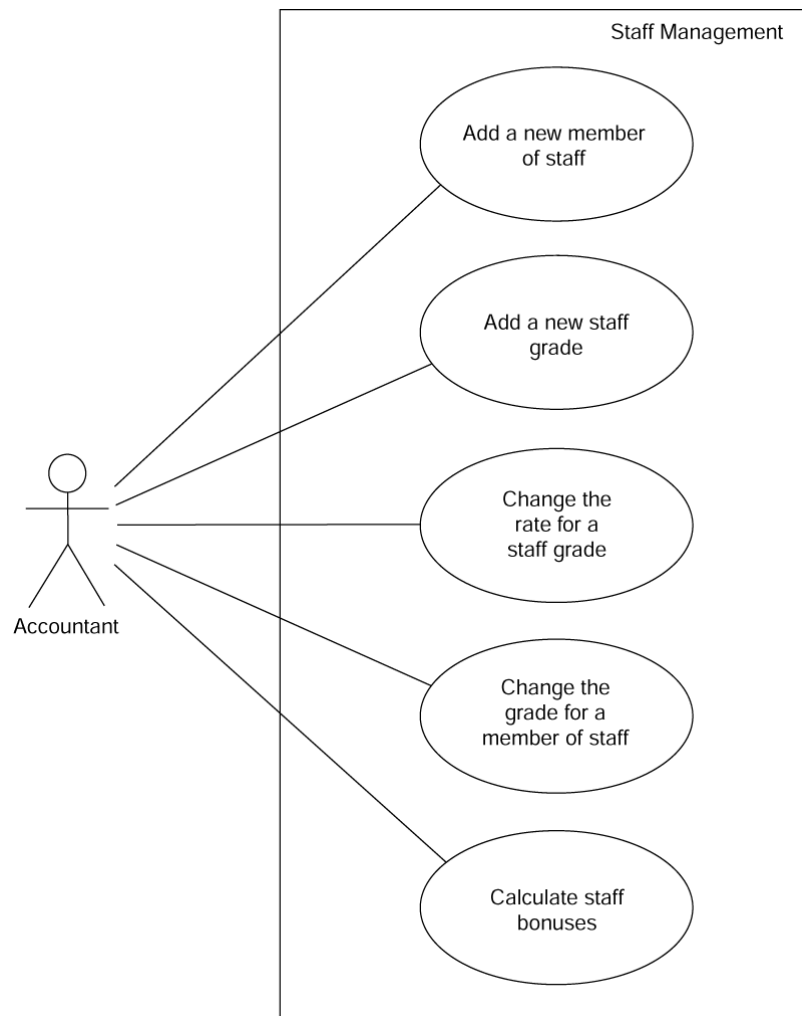
Supporting actors to the right: they provide a service.



Use Case diagrammet - notation



Eksempel på use cases



Use cases beskriver en række handlinger som systemet udfører i forbindelse med aktørens arbejdsopgaver.

Navnet på en use case skal udtrykke en handling.

Use Cases in an Iterative Process

- The UP encourages **use-case driven development** :
 - The main way to record requirements is via use cases (The Supplementary Specification is secondary) ;
 - We schedule iterations according to entire use cases (or at least use case scenarios)
 - Use cases drive the design : we try to fulfil the requirements of a use case
- Only about 10% of use cases are written in details during inception (those that are risky or architecturally significant).
- At the end of Elaboration most of the use cases are available in a detailed format; the very core of the software has also been designed and implemented.
- During the UP Construction phase, use cases need to be kept up-to-date with any changes necessary. Maybe minor remaining use case are fully detailed.

POS use cases for the inception phase:

Fully Dressed	Casual	Brief
Process Sale Handle Returns	Process rental Analyse Sales Activity Manage Security ...	Cash In Cash Out Manage Users Start Up Shut Down Manage System Tables ...