

SOFTWARE ENGINEERING

Chapter 3

Use Case Diagram



CHAPTER 3

Use Case Diagram

01

Introduction

Objective, definition and notation of the diagram

02

The elements of the diagram

Actors, use cases, and the relationships between them

03

Description of use cases

Documenting each use case: template and example

04

Example

A complete diagram, and a method to build your own

01

Introduction

What the use case diagram is for, and how it is written

Objective of the use case diagram



Build the specifications

It helps to write the document that specifies the needs of the software.



Structure the needs

It organizes the needs of the users and the corresponding objectives of the system.



Determine the functional needs

For each actor it answers two questions: Who? and What?



Identify users and interactions

It shows who the users are, and how they interact with the system.



Define the limits of the system

It draws the border between the system and its environment.

The use case diagram says WHO uses the system, and WHAT each one can do with it.

Definition



The use case diagram constitutes the first step of the UML analysis.



Model user needs

It writes down what the users expect from the system.



Identify features and limits

It shows the key functions, and where the system stops.



Represent the interactions

It links each actor to the functions he or she uses.



A user's point of view.

It describes the functionality of the system as the user sees it, not as the programmer builds it.



Main functions only.

It lists the general, essential functions without going into detail. Details come later.

Notation: the four symbols

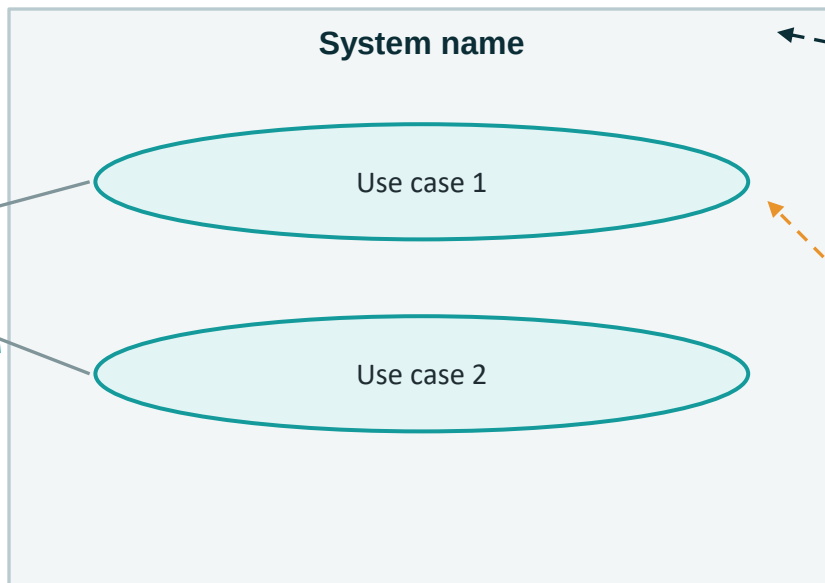
1 · Actor

a role outside the system



4 · Association

this actor uses this function



2 · System boundary

what is inside is built by us

3 · Use case

one function of the system

Four symbols are enough to draw a complete use case diagram.

02

The elements of the use case diagram

Actors, use cases, and the relationships between them

1 · The actor



An actor represents a **role** played by an external entity (human user, hardware device, or another system) that interacts directly with the system.



A human user

A customer, a librarian, an administrator...



A hardware device

A printer, a card reader, a sensor...



Another system

A payment server, a database, a web service...



An actor is a **ROLE**, not a person.

One person can play two roles (two actors); one role can be played by many people.



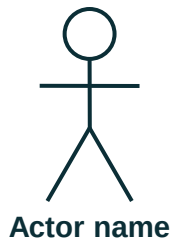
What an actor does.

It can consult and/or modify the state of the system, by sending and receiving messages that may carry data.

1.1 · How to represent actors

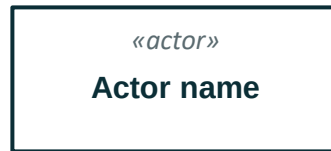
The stick man

Used when the actor is a HUMAN user.



The binder (rectangular box)

Used when the actor is HARDWARE or ANOTHER SYSTEM.



name written inside

Each actor has a name. Use a noun for the role ("Customer", "Librarian", "Payment server"), never the name of a person.

1.2 / 1.3 · Main actor and secondary actor



Main (principal) actor

The use case renders a service TO this actor. These are the people who use the main functions of the system.

Usually a human user

Customer, Member, Student



Secondary actor

Performs administrative or maintenance tasks for the system.

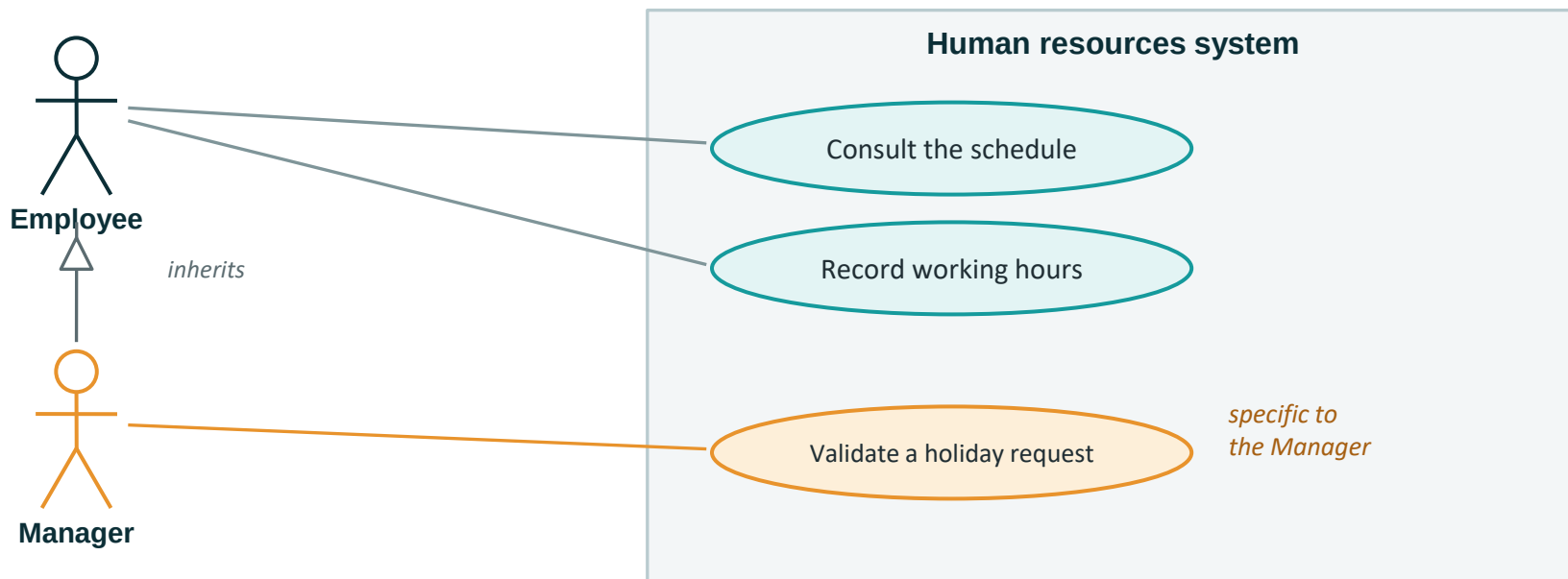
Human, hardware or other software

Administrator, Payment server, Printer

Convention of drawing: main actors are placed on the LEFT of the system boundary, secondary actors on the RIGHT.

1.4 · Generalization between actors

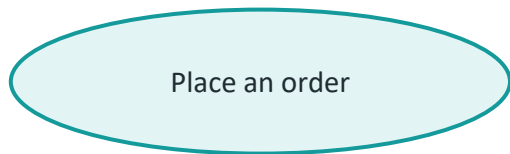
Generalization shows that an actor inherits all the use cases of another actor, and adds his own specific ones.



The Manager can also do everything the Employee can do.

2 · Use cases

A use case represents a set of **sequences of actions** performed by the system and producing an **observable result** of interest to a particular **actor**.



An ellipse containing the name of the use case.

Naming rule

A sentence starting with a VERB IN THE INFINITIVE, written from the actor's point of view.

CORRECT NAMES

✓ Place an order

✓ Search for a book

✓ Pay the invoice

TO AVOID

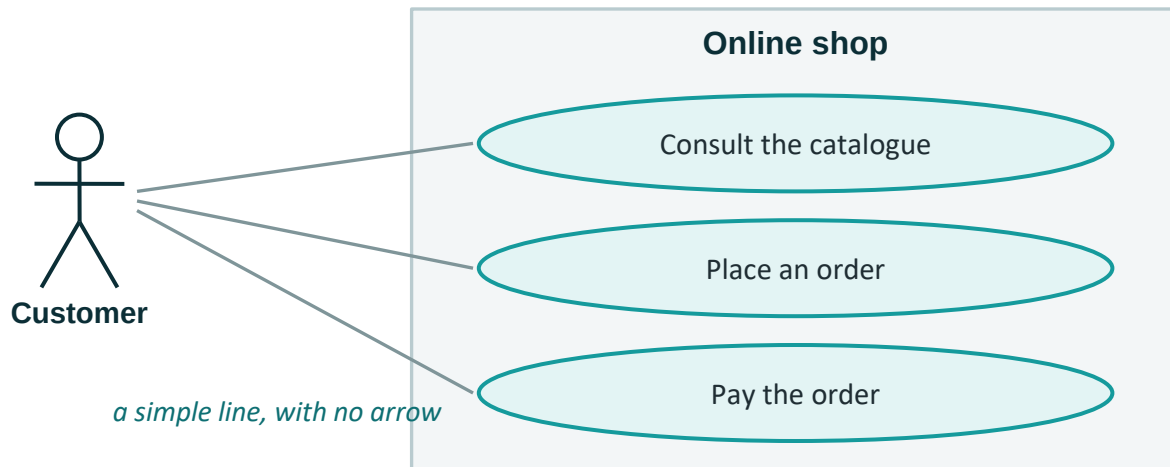
✗ Order

✗ Order form

✗ Database access

A noun alone is not a goal; a screen is not a use case.

2.1 · Association: actor ↔ use case



It means:

“this actor takes part in this function”

No arrow: the information can flow both ways.



Never link two actors together

Two actors are linked by a generalization, not by an association.



Never link two use cases with a plain line

Use «include», «extend» or generalization instead.

2.2 · The three relationships between use cases

a · Inclusion

«include»

The included case is ALWAYS executed. It is mandatory.

DIRECTION

A → B : A always includes B

b · Extension

«extend»

The extending case is executed only SOMETIMES, under a condition. It is optional.

DIRECTION

B → A : B may extend A

c · Generalization

(triangle)

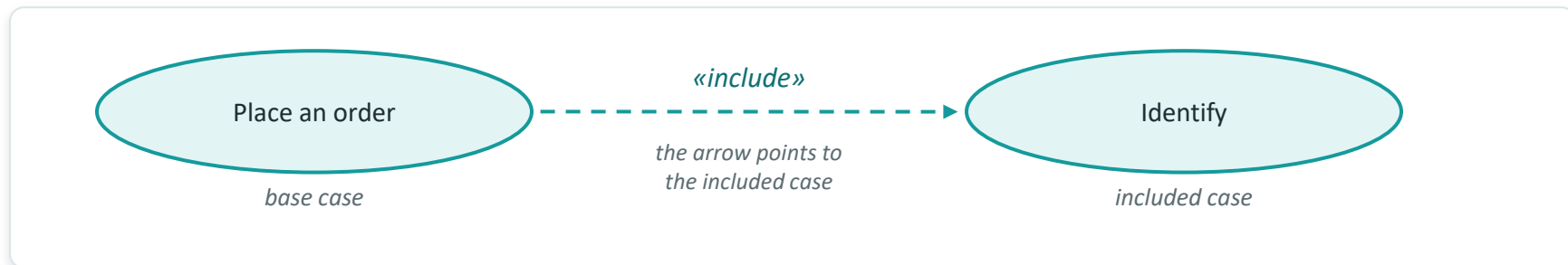
A child use case is a particular WAY of doing the parent use case.

DIRECTION

child → parent

a · Inclusion «include»

The inclusion relationship enriches one use case with another one. The included case is **always executed (mandatory)**.



To factor out

A behaviour shared by several use cases is written once and included by each of them.



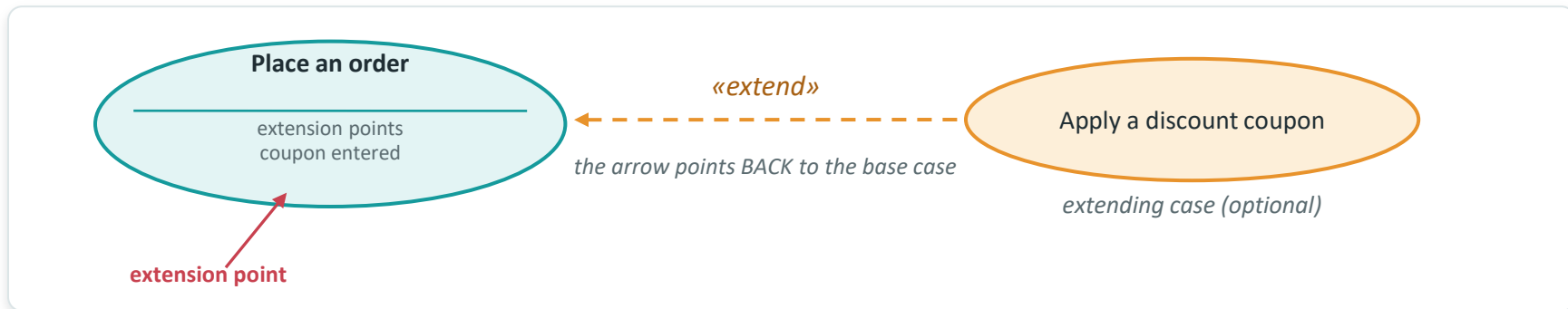
To decompose

A large use case is cut into smaller, clearer pieces.

Read it as: “to place an order, the customer must ALWAYS identify himself”.

b · Extension «extend»

The extension relationship also enriches a use case, but the extending behaviour is **optional: it happens only under a condition**.



Extension point

The named place inside the base case where the optional behaviour may be inserted.



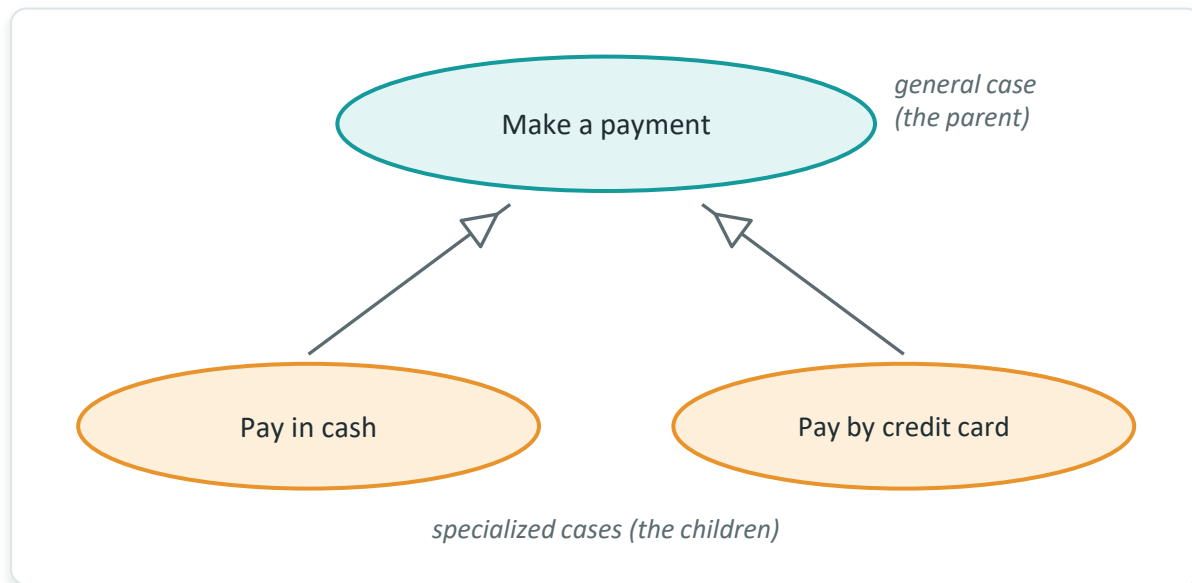
Condition

The extension happens only if the condition is true ("if the customer has a coupon").

Read it as: "while placing an order, the customer MAY apply a discount coupon".

c · Generalization / specialization

A child use case is a particular WAY of carrying out the parent use case. The triangle points to the general case.



The child inherits the behaviour of the parent, and modifies or completes it.

Notation

A solid line with a hollow triangular end. The triangle is on the side of the general case.

When to use it

When several use cases have the same goal, but reach it by clearly different means.

«include» vs «extend»: do not confuse them

	«include»	«extend»
Is it executed?	Always — mandatory	Only sometimes — optional
Condition	No condition	Under a condition
Direction of the arrow	base → included	extending → base
Is the base case complete?	No: it needs the included case	Yes: it works without the extension
Typical use	Factor out a shared behaviour	Add an exceptional or optional behaviour
Example	Place an order «include» Identify oneself	Apply a coupon «extend» Place an order

Summary of the notation



Human actor

a role played by a person



Non-human actor

a device or another system



Use case

one function of the system



System boundary

what we are going to build



Association

this actor uses this function



«include»

mandatory: always executed



«extend»

optional: executed under a condition



Generalization

inheritance between actors or use cases

03

Description of use cases

Documenting each use case in text, so that nothing stays ambiguous

Why describe each use case?



The diagram shows the main functions, but it does NOT detail the dialogue between the actors and the use cases. An ellipse named “Place an order” can be understood in several different ways.

Each use case must be documented so that there is no ambiguity about how it works and what exactly it covers.



1 · Identification

Name, actors, goal, pre-conditions and post-conditions of the use case.



2 · Description of the scenarios

The main scenario, step by step, and the alternative scenarios.



3 · Non-functional requirements

Performance, security, usability, and optional screen mock-ups.

The description template

Field	What it contains
Use case	The name of the use case.
Main actor	The main (principal) actors.
Secondary actor	The secondary actors.
Goal	What the use case is for.
Pre-condition	What must be true BEFORE the use case can start.
Post-condition	What must be true WHEN the use case ends.
Main scenario	The one that satisfies the actors' objectives through the most direct path to success.
Alternative scenario	All the other scenarios, whether successful (normal completion) or failed (error).
Additional requirements	Performance, security, usability... The scenarios may be completed with screenshots of the prototype.

The three blue rows are the heart of the description: they tell the complete story of the use case.

Example: “Add an order”

Field	Content
Actor	Customer.
Objective	Add an order.
Pre-condition	The use case begins when the user is authenticated.
Post-condition	The customer has placed an order.
Nominal scenario	1. The customer asks to fill in the order form. 2. The system displays the form. 3. The customer enters the necessary information (surname, first name, address, etc.). 4. The system records the information and returns a success message.
Error scenario (alternative)	4-a. The customer has not filled in a required field, or has entered invalid information. The system shows the unaccepted field in red, and the scenario resumes from step 2.
Additional requirements	/

Note the numbering “4-a”: the alternative starts at step 4 of the nominal scenario, and says where the story continues.

Main scenario and alternative scenarios



Main (nominal) scenario

- Everything goes well: the ideal story
- The most direct path to success
- Written as a numbered list of steps
- Alternate: the actor acts → the system answers



Alternative scenarios

- All the other stories
- Successful ones: another normal way to finish
- Failed ones: an error or a refusal
- Numbered from the step where they branch off

HOW TO NUMBER AN ALTERNATIVE

4-a. the step it branches from → what happens → where the scenario continues (“resumes from step 2”, or “the use case ends in failure”).

How to write a good description



One short sentence per step

Active voice, and always name who acts: “The customer enters...”, “The system displays...”.



No buttons, no windows

Stay independent from the interface: describe the goal, not the clicks.



Number every step

The alternative scenarios refer to these numbers.



Between 3 and 10 steps

If it is longer, the use case should probably be cut into several ones.



The customer's vocabulary

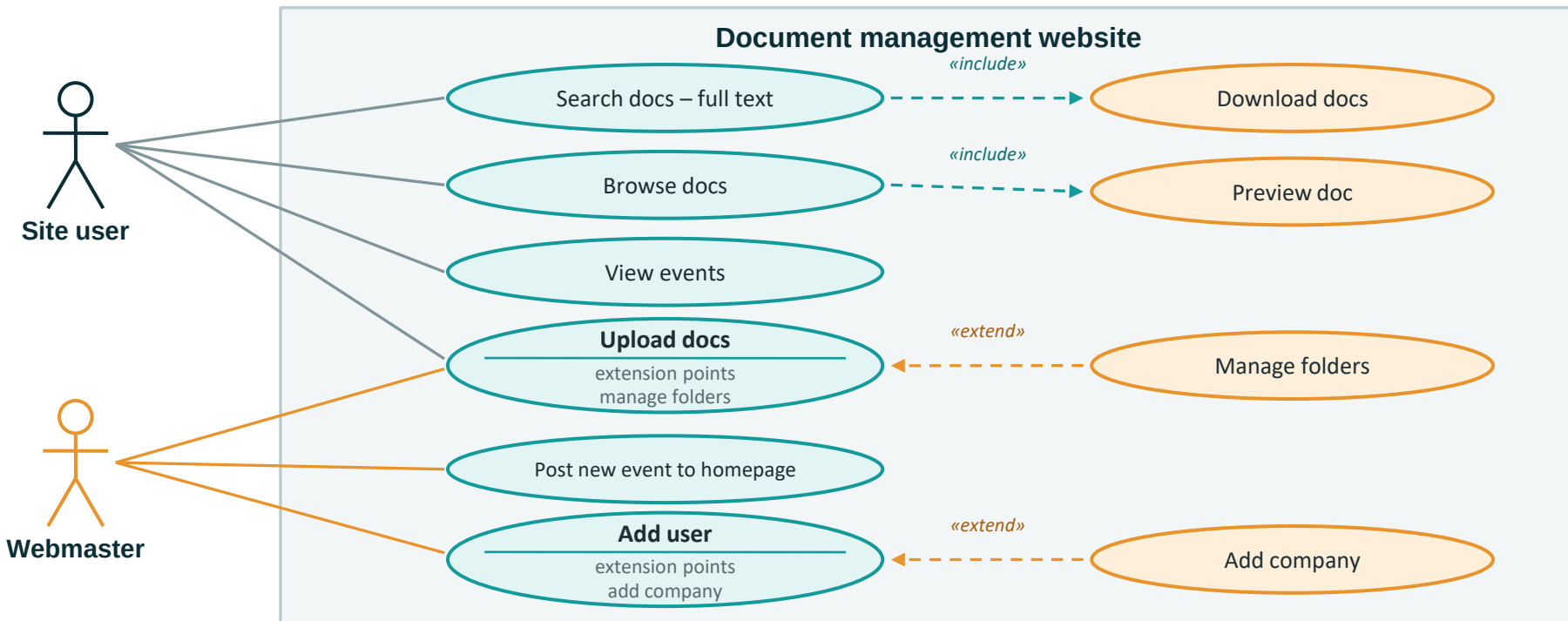
No technical words: the customer must be able to read and validate it.

04

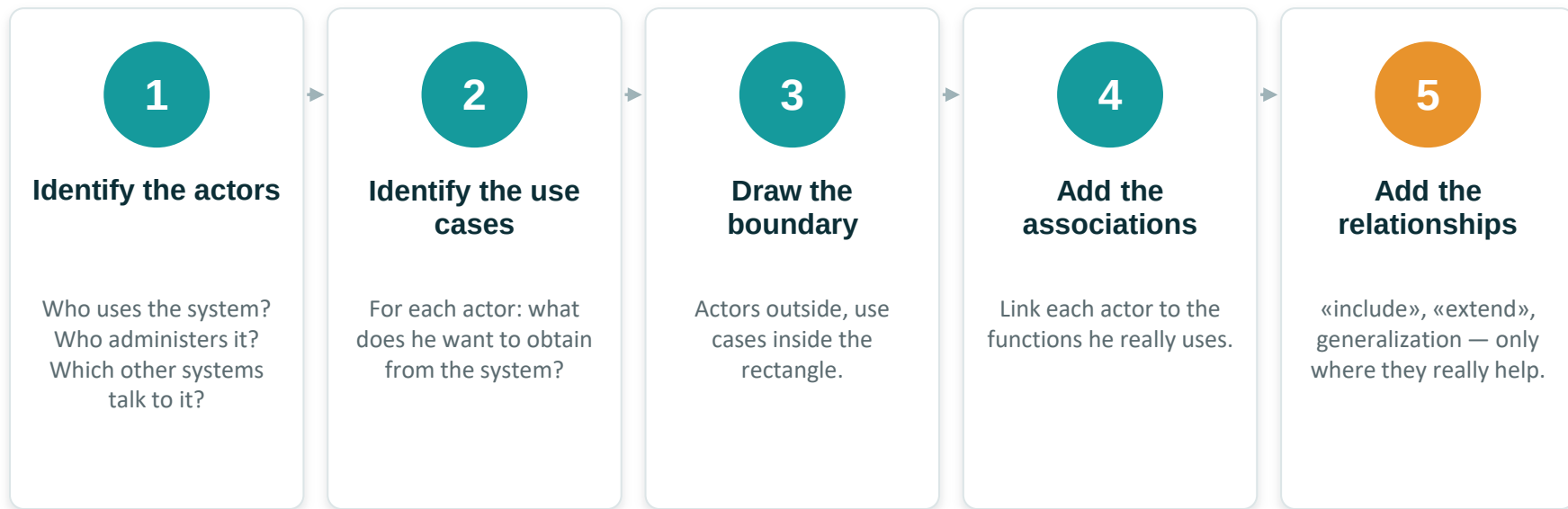
Example

A complete diagram, and a method to build your own

A complete example: a document website



Method: building a diagram in 5 steps



Do not start with step 5: a diagram covered with «include» and «extend» arrows becomes unreadable.

A cinema booking system

THE STATEMENT

- A cinema wants a seat booking website.
- A visitor can consult the programme and search for a film.
- To book a seat, the visitor must create an account, then log in.
- While booking, he may choose a seat on the plan of the room, and he may apply a promotional code.
- Payment is made either by credit card or at the counter.
- The manager updates the programme and consults the statistics.

HINTS

2 actors

Visitor and Manager

«include»

“Book a seat” always includes “Log in”

«extend»

“Choose a seat” and “Apply a promotional code” are optional

Generalization

“Pay by card” / “Pay at the counter”
specialize “Pay”

Then write the textual description of the use case “Book a seat”, using the template of part 03.

KEY TAKEAWAYS

What to remember from Chapter 3



The first step of UML analysis: WHO uses the system, and WHAT for.



Four symbols only: actor, use case, system boundary, association.



An actor is a ROLE: a human, a device, or another system.



A use case is named with a verb in the infinitive: “Place an order”.



«include» = always · «extend» = sometimes — opposite arrow directions.



The diagram is not enough: describe each use case in text.

Bibliography



UML 2 — pratique de la modélisation

Benoît Charroux, Yann Thierry-Mieg, Aomar Osmani

[Open link](#)



UML 2 par la pratique

Pascal Roques · Éditions Eyrolles



Les cahiers du programmeur — UML 2

Pascal Roques · Éditions Eyrolles



UML en action

Pascal Roques · Éditions Eyrolles — complete case studies



OMG Unified Modeling Language — the official specification

Object Management Group

[Open link](#)