Software Architecture and Design I
SOEN 343

Instructor: Dr. Rodrigo Morales
https://moar82.github.io/
rodrigo.moralesalvarado@concordia.ca

Lecture 3a. Use Cases

Learning objectives (1)

- Understand the role of use cases for requirements specification
- Understand the main benefits of use cases, compared with other specification forms
- Understand the main concepts behind use cases, including actors, goals, and scenarios
- Understand how use cases are iteratively refined (and implemented) in an UP project

Learning objectives (2)

- Understand how to systematically build a use case model
- Understand how to find actors and learn the difference between primary and secondary actors
- Understand the concept of goals and how they relate to actors and use cases
- Learn how to identify different goal levels, in particular the user-goal and subfunction-goal levels
- Understand how to separate the main success scenario from other scenarios
- Learn about UML use case diagrams and use case context diagrams

Learning objectives (3)

- Learn how to refine use cases into “fully-dressed” specifications
- Learn about subfunction goal-level use cases and how they relate to user-goal level use cases
- Learn about the «extend» and «include» relations between use cases
- Learn how to define pre- and post-conditions in use cases
- Learn how to describe stakeholders’ interests within a use case
- Learn how to connect use cases with non-functional requirements
- Learn about writing rules and checklists for use case specifications
- Understand how use cases are handled in design and implementation in the UP

Introduction

- While it is true that not everything that matters about requirements is a use case, it is also true that the use case will be the requirement's workhorse
- A use case describe sequences of actions a system perform that yield an observable result of value to a particular actor

Use cases: Features

- Are part of the SRS
- Contain **actors**
- Describe an observable value delivered by the system to an actor
- Capture functional requirements
- Are derived from the *Vision Document* in the UP process
- Collect a number of (positive) scenarios, both normal and abnormal

Why Use Cases?

- “Old-style ” SRS just have long list of requirements structured by sub-system:

```
2.1 Data Entry
2.1.1 Data Entry Subsystem
2.1.1.1 User shall login to system
...
2.2 Report Generation
...
2.2.4.2.1 Reports must be sortable by due date or entry date
```

- Commonly associated with Waterfall model

Real life example: NASA

WIRE C&DH Flight Software Requirements Specification

041.1.1 The flight software shall receive data in the form of CCSDS packets from the Software Bus and shall transmit the data over the 1553B bus.

041.1.2 The flight software shall collect data from the 1553B remote terminals and shall transmit the data on the Software Bus in the form of CCSDS packets.

041.1.3 If necessary, the flight software shall use multiple 1553B transactions to collect or transmit a single CCSDS packet.

https://web.archive.org/web/20111015090659/http://sunland.gsfc.nasa.gov/smex/wire/mission/cdhsw/wirrqtop.htm

The benefits of use cases

- Compared to traditional requirements methods, use cases are relatively easy to write and easier to read
- Use case force developers to think through the design of a system from the perspective of a user
- They provide an ordering mechanism for requirements
- They are critical tool in the analysis, design, implementation and testing process
- They also serve as inputs to the user documentation, conveniently organized in a step-by-step format

Use cases elements

- **Actor**: something with behavior, such as a person, computer system, or organization, e.g., a cashier
- **Scenario**: a specific sequence of actions and interactions between actors and the system under discussion, 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

Use case example

UC: Handle returns

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 credit reimbursement authorization is rejected, inform customer and ask for an alternative payment method.
- If item identifier not found in the system, notify the Cashier and suggest manual entry of the identifier code.
- …

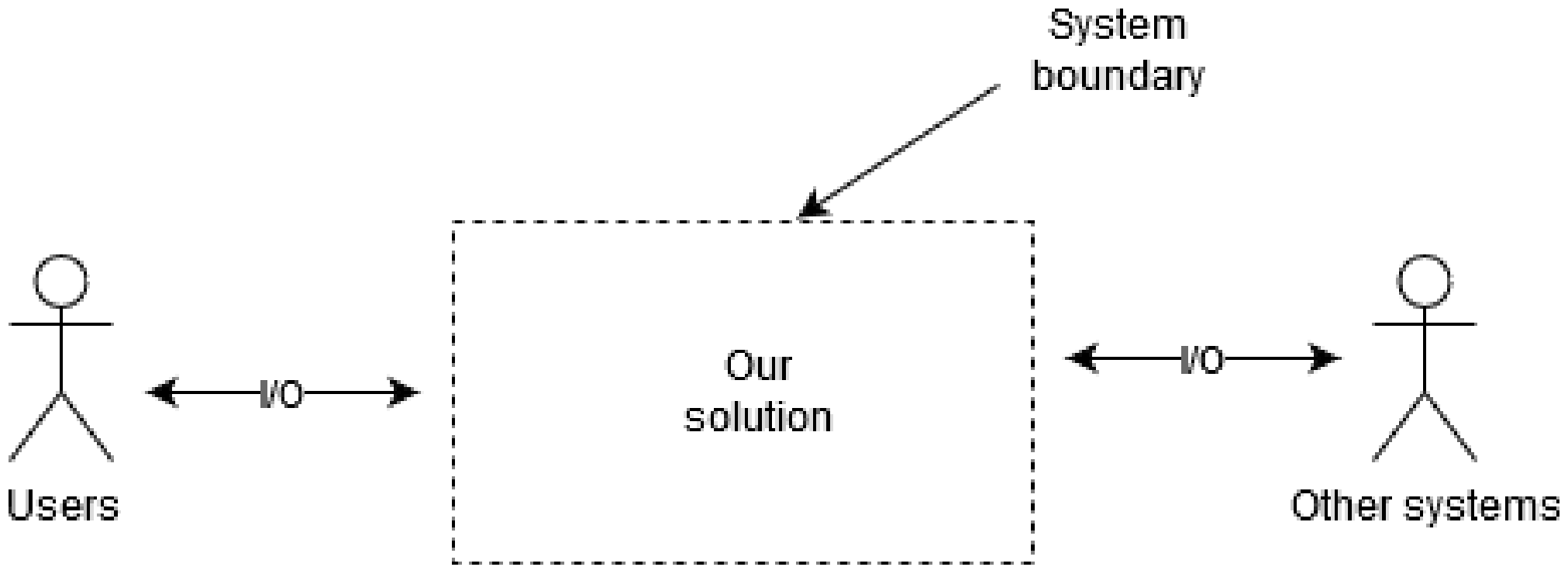
Building the Use Case Model step by step

1. Identify and describe the Actors
2. Identify the Use Cases and write a Brief Description
3. Identify the Actor/Use Case Relationships
4. Outline the Individual Use Cases
5. Refine the Use Cases

Building the Use Case Model step by step

1. **Identify and describe the Actors**
2. Identify the Use Cases and write a Brief Description
3. Identify the Actor/Use Case Relationships
4. Outline the Individual Use Cases
5. Refine the Use Cases

System Boundary is the context



How to identify Actors

- Who will supply, use, or remove information from the system?
- Who will operate the system?
- Who will perform any system maintenance?
- Where will the system be used?
- Where does the system get its information?
- What other external systems will interact with the system?

Primary vs Secondary actors

Primary actor:

- Interacts to achieve required system function and derives the intended benefit from the system
- Works directly and frequently with the system
- Usually positioned on the **left** side of the Use case diagram

Secondary actor:

- Supports the system so that primary actors can do their job
- Usually positioned on the **right** side of the Use case diagram

Building the Use Case Model step by step

1. Identify and describe the Actors
2. **Identify the Use Cases and write a Brief Description**
3. Identify the Actor/Use Case Relationships
4. Outline the Individual Use Cases
5. Refine the Use Cases

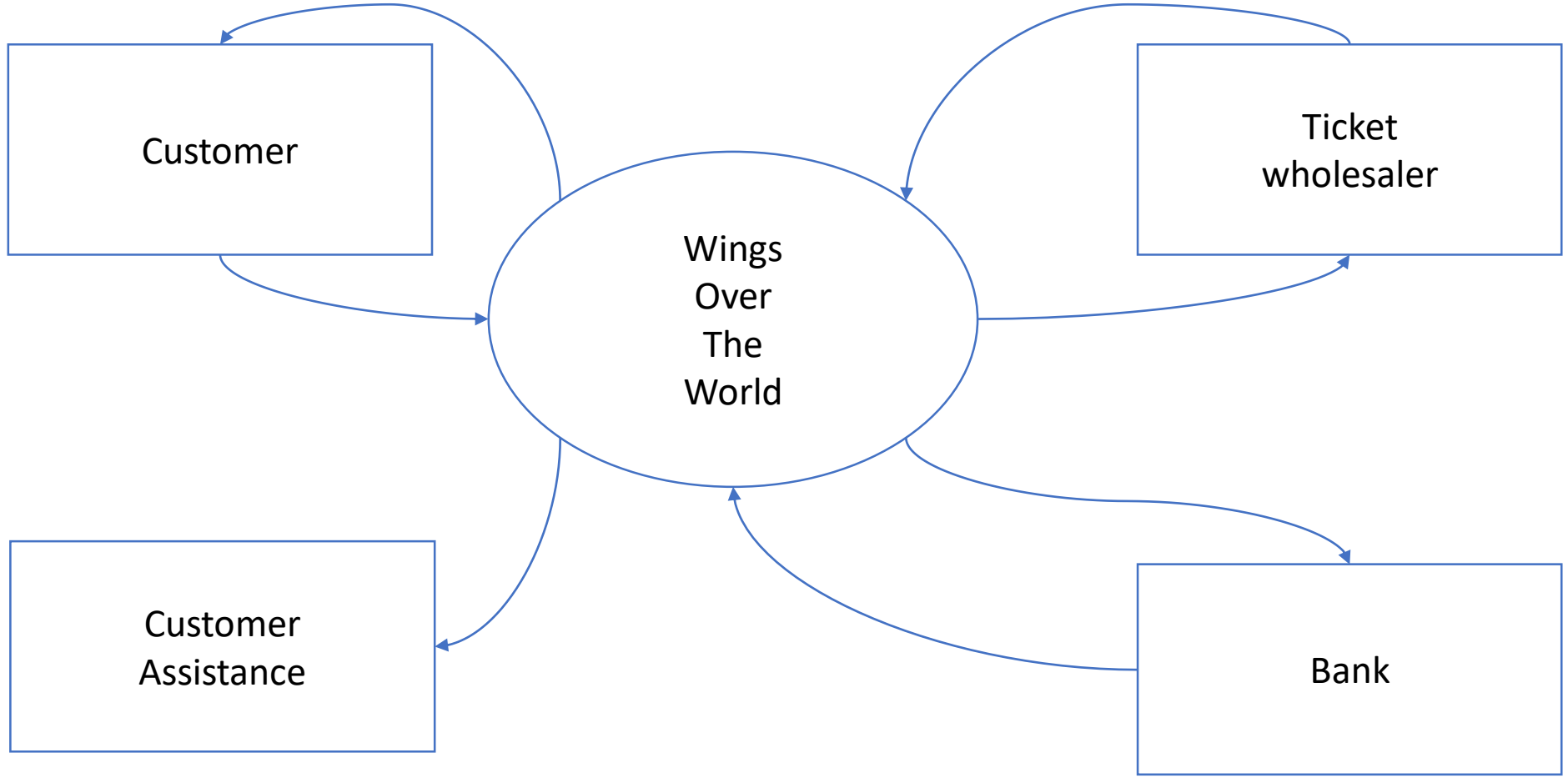
Example: Wings over the world company

Vision (Objectives and Goals)

Wings Over the World endeavors to maintain its reputation for innovation by increasing access to our travel services and by offering new and innovative services that are unmatched by our competitors. Specifically:

- Increase brand awareness of Wings Over the World with the creation of a public web site
- Increase market share by 15 percent and lower the cost of booking tickets by letting 30 percent of our clients to book on-line.
- Open the Wings Over the World travel system to independent travel agents by offering premium booking services and therefore create a new revenue stream

Context diagram (not UML)



User-Goal Level use cases

- Satisfies a particular and immediate goal of value to the primary actor

Examples:

- Wings over the world user-level use cases:
  - Book Flight
  - Book Hotel
- Writing Rule: Name the use case with an **active verb phrase** that represents the goal of the primary actor
- Use Case Brief: A simple, one-paragraph story describing the main success scenario for the use case

Example: Book Flight

UC: Book Flight

Actor: Travel agent

The travel agent retrieves a client's reservation and books the flight. The agent examines the aircraft seat map and selects the client's preferred seats in the aircraft. The agent enters the client's payment information, and the system books and assigns the seats. The system prints tickets

Building the Use Case Model step by step

1. Identify and describe the Actors
2. Identify the Use Cases and write a Brief Description
3. **Identify the Actor/Use Case Relationships**
4. Outline the Individual Use Cases
5. Refine the Use Cases

How to do it?

- First identify the actors and their goals
- Actors: What computers, subsystems and people will drive our system?
- Goals: What does each actor need our system to do?
  - Each goal shows up as a trigger to a usage (use case) of our system.
- Result: A list of use cases, a sketch of the system
  - Short, almost complete list of usable system function

Wings Over the World: Actors and Goals

Actor	Goals
Agent	Reserve flight Book flight Cancel flight reservation Request upgrade Open passenger profile Close passenger profile
Airline	Cancel flight Discount flight

Actor-Goal list

Initial List

Actor	Goal (use case)
User	Login Submit report
Administrator	Login Submit report Add Users Remove Users …

Actor-Goal list

Refined List (User-goal level)

Actor	Goal (use case)
User	~~Login~~ Submit report
Administrator	~~Login~~ Submit report ~~Add~~ Manage Users ~~Remove Users~~ …

The boss test

- Your boss asks, "*What have you been doing all day*?'
- You reply: "*Logging in!*"
- Is your boss happy?
- **If not**, your use case failed the Boss test (as it does not deliver measurable value to the user)

The EBP Test

- Check if UC is an Elementary Business Process (EBP)
- *A task performed by one person in one place at one time, in response to a business event, which adds measurable business value and leaves the data in a consistent state, e.g., Approve Credit*
- Don't confuse use cases with "functions" (like entries in a menu – add order, delete order, update order, …)

The size test

- Typically a UC has many steps
- In “fully dressed” format, can be elaborated to 3-10 pages or more
- If you end up with only 1-2 steps, this is most likely not a valid use case

Wings Over the World: Actors and Goals

Actor	Goals
Agent	Reserve flight Book flight Cancel flight reservation Request upgrade ~~Open passenger profile~~ ~~Close passenger profile~~
Airline	Cancel flight Discount flight

Exercises to reflect

Test the following use cases:

- Negotiate a strategic company alliance
- Handle returns
- Log in
- Update piece on game board

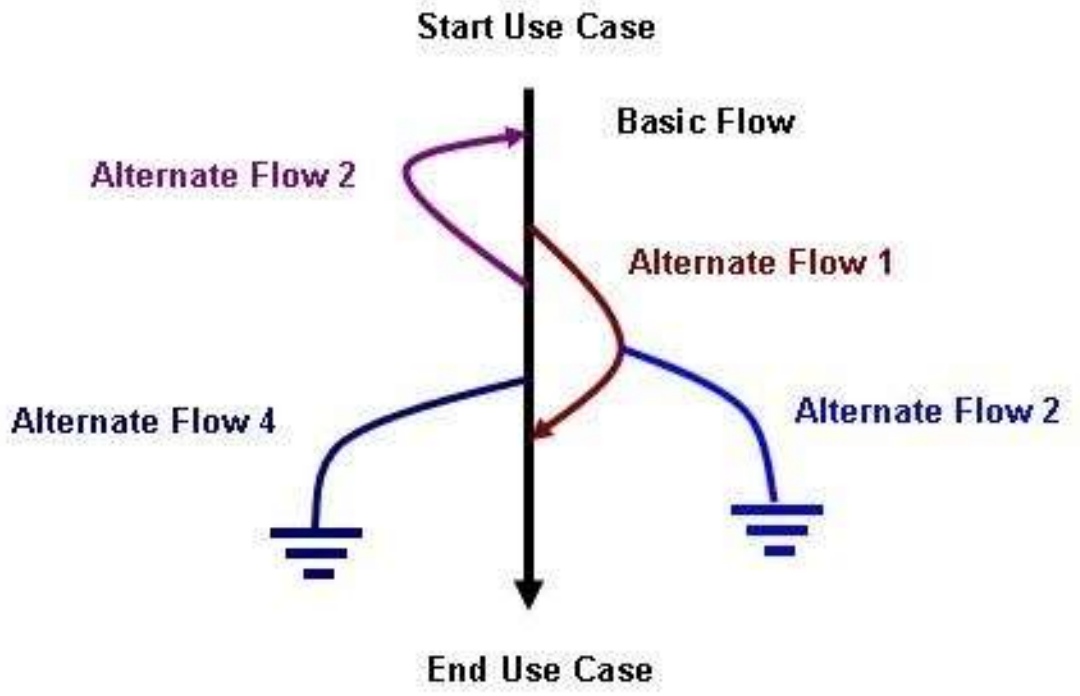
Building the Use Case Model step by step

1. Identify and describe the Actors
2. Identify the Use Cases and write a Brief Description
3. Identify the Actor/Use Case Relationships
4. **Outline the Individual Use Cases**
5. Refine the Use Cases

Outline the individual Use cases

Outline the basic flow first (black line)

- Basic flow
  - What event starts the use case?
  - How does the use case end?
  - How does the use case repeat some behavior?
- Alternate flow:
  - Are there optional situations in the use case?
  - What odd cases might happen?
  - What may go wrong?
  - What may not happen?
  - What kind of resources can be blocked?



User goal Main Success Scenario

- **UC:** Book Flight
- **Level:** User Goal
- **Main Success Scenario**

1. This use case begins when a customer calls and requests a flight
2. The customer describes her flight needs by specifying her origination, destination, travel dates and preferred departure times
3. The system looks up all flights that match the customer's travel preferences and presents the travel options to the customer
4. The customer selects a flight
5. The system builds a flight itinerary for the customer.
6. The system reserves the flight for the customer
7. The customer provides a credit card number and charges the price of the flight against it
8. The system issues the ticket to the customer

Some use case writing rules

- Each scenario begins with a **triggering action** that the system can detect:
  - ATM Withdraw Cash Main Success Scenario
    1. Customer inserts the card.
    2. …
- Steps should not be too small
  - Stepping up to the sidewalk Main Success scenario
    1. She lifts her foot from the street to the curb.
    2. She arcs the foot through the air, and lowers to the pavement
- Keep scenarios to three to nine steps, all at a level of abstraction just below the use case goal

Essential vs Concrete style

Essential

- Focus on intent: Avoid making UI decisions

Concrete

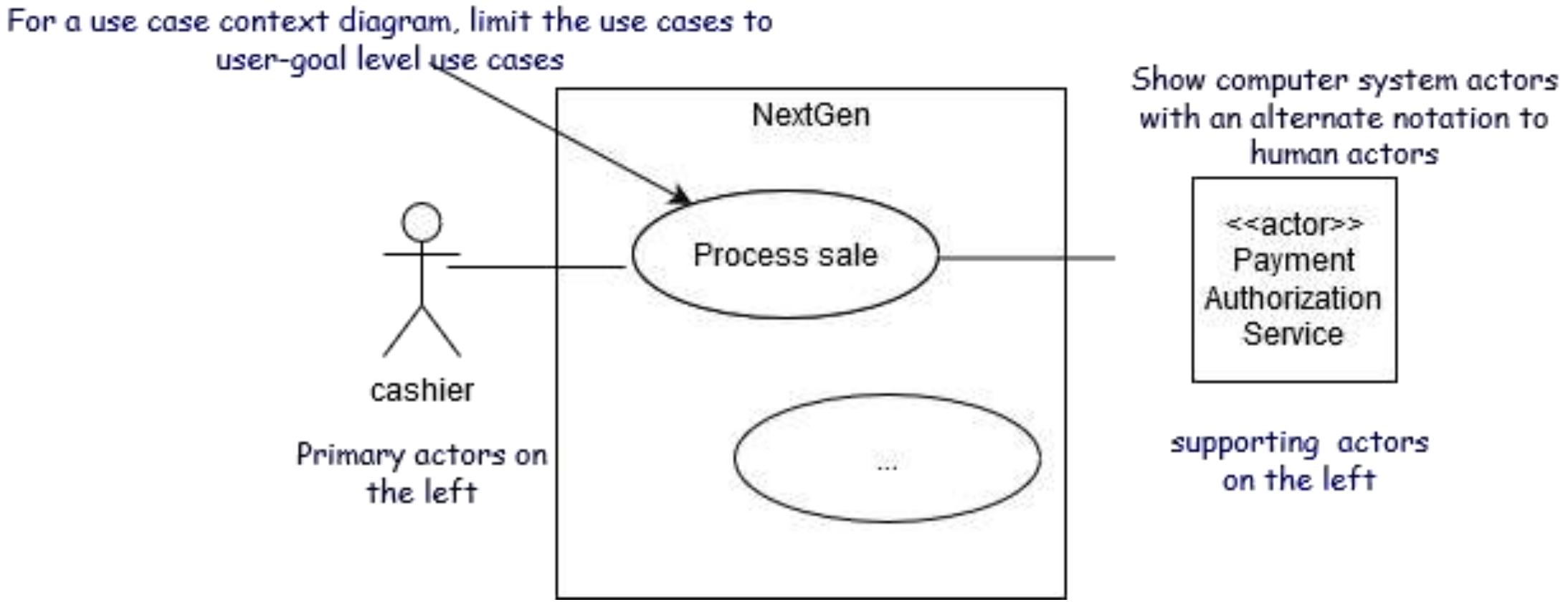
- UI decisions are embedded in the use case text: “Admin enters ID and password in the dialog box (see picture X)”

Concrete style not suitable during early requirements analysis work

UML and the use case model

- Use case diagram
  - UML defines the Use case diagram, but doesn't explain how write actual use cases!
- Use cases are **text documents**, **not diagrams**
- Use-case modeling is primarily an act of **writing text**, not **drawing diagrams**

Use case Diagram as a Context Diagram: POS



Relating Use cases

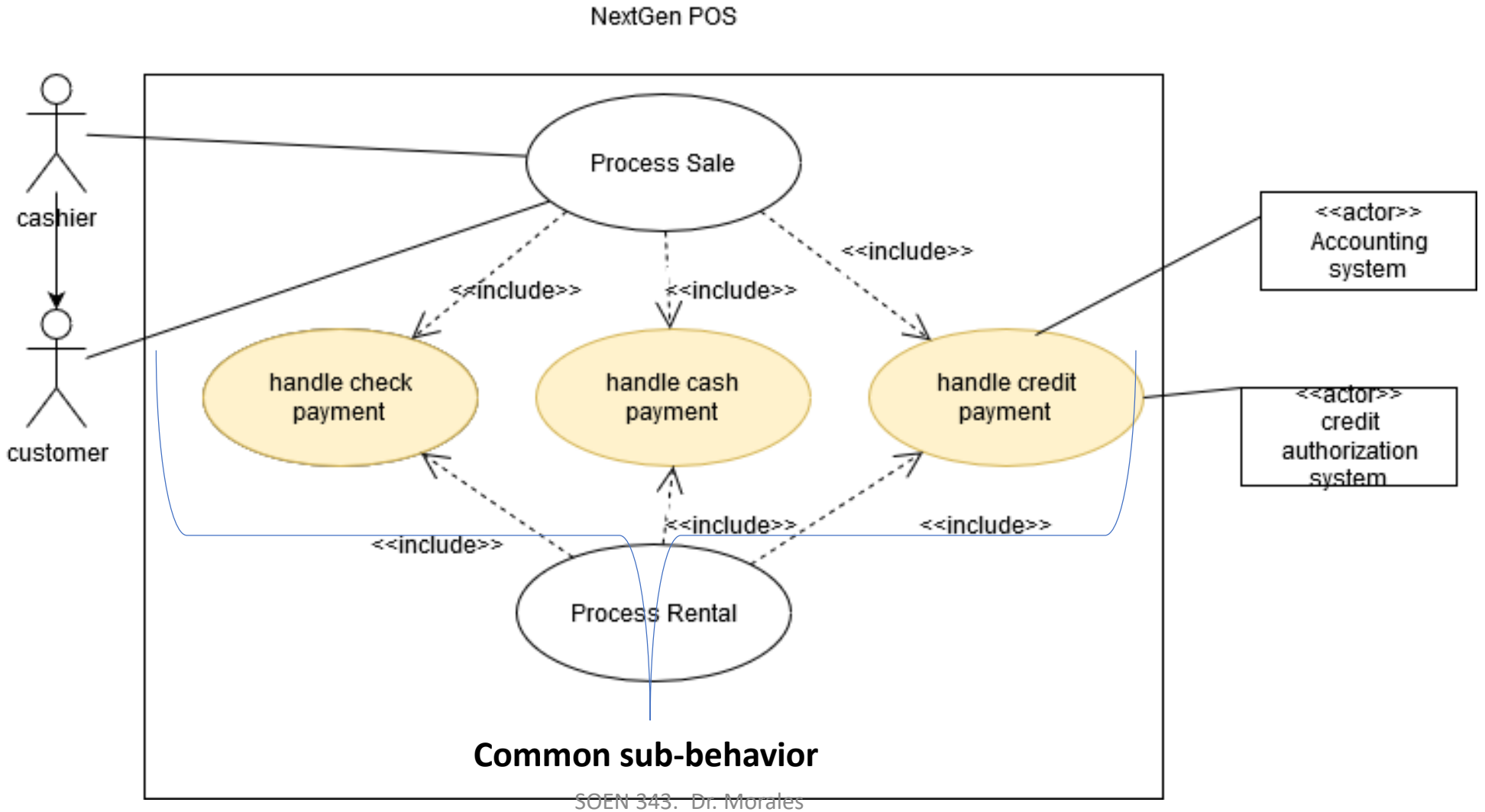
The include relationship

- partial behavior that is common across several use cases (models required behavior)
- separate it into its own Subfunction Use Case, and indicate its inclusion
- Included sub-function shared across several use-cases (duplicated parts!)

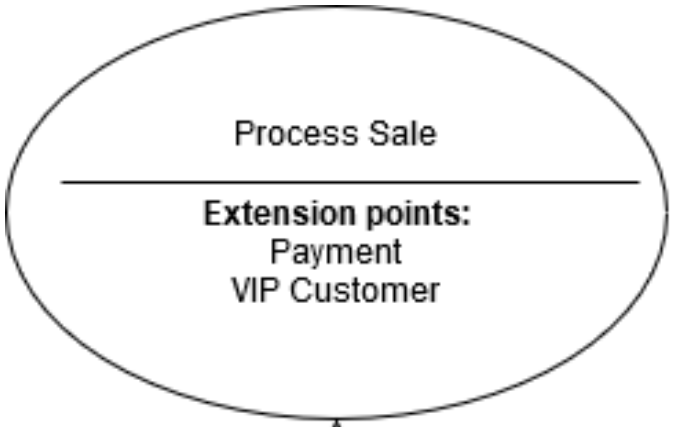
The extend relationship

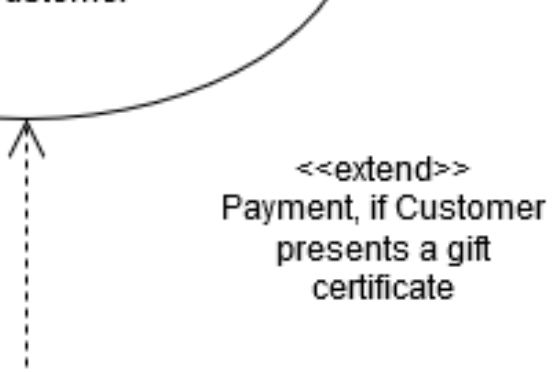
- separate it into its own Subfunction Use Case
- indicate where and under what condition it extends the behavior of some base use case (it models optional behavior)
- Original use case remains untouched
- Extended functionality may be optional

UML: Using «include» Relationship



UML: Using Extend Relationship





UML notation:

1. The extending use case points to the base use case
2. The condition and extension point can be shown on the line

Common sub-behavior

Building the Use Case Model step by step

1. Identify and describe the Actors
2. Identify the Use Cases and write a Brief Description
3. Identify the Actor/Use Case Relationships
4. Outline the Individual Use Cases
5. **Refine the Use Cases**

Refine use cases

- At some point later in the project lifecycle, the time will be right to refine the use cases to the next and last level of detail
- *All alternate flows, including exception conditions*
  - "What if the remote server is down?“
  - All these exceptions must be documented in the use case or the application may not behave as expected
- *Pre- and post-conditions*
  - A pre-condition to programming vacation settings might be that the user has set the calendar clock
  - Post-condition example: the programmed vacation schedule, which reflects that actual input by the homeowner, must be saved by the system to be recalled for later use.

Process Sale: User Level Use Case Example

Preconditions:

- Cashier is identified and authenticated.

Success Guarantee (Postconditions):

- Sale is saved.
- Tax is correctly calculated.
- Accounting and Inventory are updated.
- Commissions recorded.
- Receipt is generated.
- Payment authorization approvals are recorded.

Fully dressed use cases template

Use case section	Comment
Use Case Name	Start with a verb
Scope	The system under design
Level	“User-level” or “subfunction”
Primary Actor	Calls on the system to deliver its services.
Stakeholder and interests	Who cares about this use case, and what do they want?
Preconditions	What must be true on start, and worth telling the reader?
Success Guarantee	What must be true on successful completion, and worth telling reader?
Main Success Scenario	A typical, unconditional happy path scenario of success.
Extensions	Alternate scenarios of success or failure.
Special Requirements	Related NFRs
Technology and Data Variations list	Varying I/O methods and data formats.
Frequency of Occurrence	Influences investigation, testing, and timing of implementation.
Misc.	Such as open issues.

More use case writing rules

Use cases do not collect formulas, state, cardinality

- Bad examples:
  - Order sum = order item costs * 1.06 tax. (design issue)
  - Promotions may not run longer than 6 months. (business rule)
  - Customers only become ‘Preferred’ after an initial 6-month period. (business rule)

Write in an UI-free style

- Bad examples:
  - Systems displays the “edit users” window (see Figure 1). (design issue)

Focus on intent

- Bad examples:
  - Administrator enters ID and password in dialog box.
  - (Better: 1. **Administrator identifies self.**)

Conclusion

- **Writing good use cases is a creative task**
- You are describing a systems that does not yet exist!
- This requires vision and creativity
- Don’t start to analyze structure or implementation
- Synthesize instead of Analyze!
- Take time to understand the domain
- Understand the software development involved
- Assign people to writing Use Cases with good technical writing skills

References

1. Chapter 14, 21 and appendix C., Dean Leffingwell, Don Widrig: *Managing Software Requirements: A Use Case Approach*. 2nd ed., Addison-Wesley, 2003.