CSC 4104

# *Information System -Requirements and Use Case Scenarios*


## *Dr J Paul Gibson*

**Dept. INF**

**Office D311**

**paul.gibson@telecom-sudparis.eu**


**http://jpaulgibson.synology.me/~jpaulgibson/TSP/Teaching/CSC4104/CSC4104-InformationSystem-RequirementsUseCases.pdf**

# Additional Reading Material

*Requirements engineering in the year 00: A research perspective*, A van Lamsweerde, 2000 pdf

*Requirements Engineering: A Roadmap*, Bashar Nuseibeh and Steve Easterbrook, 2000 pdf

*On Non-Functional Requirements in Software Engineering*, Lawrence Chung and Julio Cesar Sampaio do Prado Leite, 2009 pdf

*Requirements Engineering*, Elizabeth Hull, Ken Jackson and Jeremy Dick, 2005 html

*Use cases - yesterday, today, and tomorrow*, Ivar Jacobson, 2004 pdf

*Structuring Use Cases with Goals*, Alistair Cockburn, 1997 pdf

*Writing effective use cases. Vol. 1*, Alistair Cockburn, 2000 pdf

**Use case modeling** is process of describing behavior of system from external point of view

– Use case describes **what** a system does, **not how** it does it

– Emphasizes modeling **external**, not internal, **point of view** to focus on requirements, not implementation

– Captures requirements of system as list of **structured scenarios**

– Use cases are the basic unit of **requirements** definition

– **Actor** in use case can be person, computer/device or external system

– Actor represents group of users or **role**, not specific individual

## What is a use case?

Use cases are descriptions of the **ways users interact** with systems to accomplish tasks or reach goals. Mapping these interactions can improve early planning and ensure a smooth development cycle.

A use case explains **how users interact** with a product or system. It outlines the flow of user inputs, establishing successful and failed paths to meeting goals

## What is a use case?

Use cases vary in complexity depending on your audience or system. But across the board, your use case should identify a few key components. The most important ones include:

- **Actor**: anything exhibiting behavior that interacts with a system, such as a single user, a team, or another piece of software
- **System**: the product or service with defined functionality
- **Goal**: the purpose or objective users reach with a system's features

## Other Use Case Elements

CSC 4104

Actors, systems, and goals build the foundation for a use case.

When you begin tracking system interactions, a few new elements come into play:

- **Stakeholder**(s): someone with a stake or interest in a system's performance
- **Primary actor**: the actor who initiates a system's function to reach a goal
- **Preconditions**: underlying factors required for the use case to happen
- **Triggers**: events that begin a use case
- **Basic flows**: use cases where systems work as intended to reach a goal
- **Alternate flows**: different outcomes based on when and how a system veers off course

# Types of use cases

Use cases come in two forms: **business** and **system**.

A system use case is a detailed look at how users interact with each part of a system. It highlights how unique inputs and contexts cause the system to reach different outcomes. This level of detail highlights how a system's individual functions work in any scenario.

Business use cases paint a more general picture of how a user might interact with your business to reach their goals. Instead of focusing on technical detail, it's a cause-and-effect description of different inputs. For example, if you run a code debugging platform, your business use case explains how users enter their code and receive error notices.

## Business and/or System?

Some teams like to write a business use case to outline a system's processes before development.

As developers begin their work, a manager will outline more technical system use cases to follow.

## Use scenario vs. use case

Use cases show all the ways a system functions when trying to reach goals, but a scenario only depicts one example.

In a scenario, the system can succeed or fail at reaching the user's goals.

Put simply, multiple use scenarios build a use case.

# Use case vs. user story

Use cases depict how users interact with a system, and user stories describe features from the user’s perspective.

As a result, user stories are much shorter than use cases, typically consisting of brief descriptions teams use as a jumping-off point in development.

Additionally, use cases can assist multiple teams in an organization, while user stories help product teams build their tool.

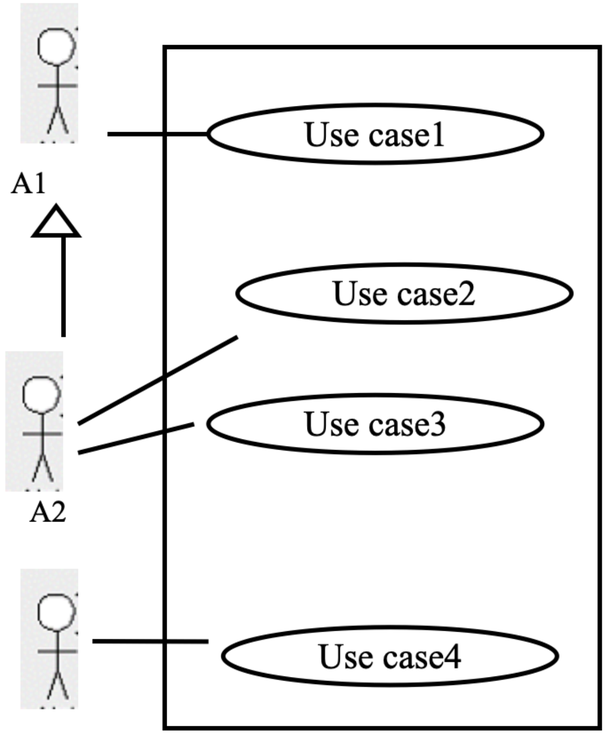
## Use case vs. test case

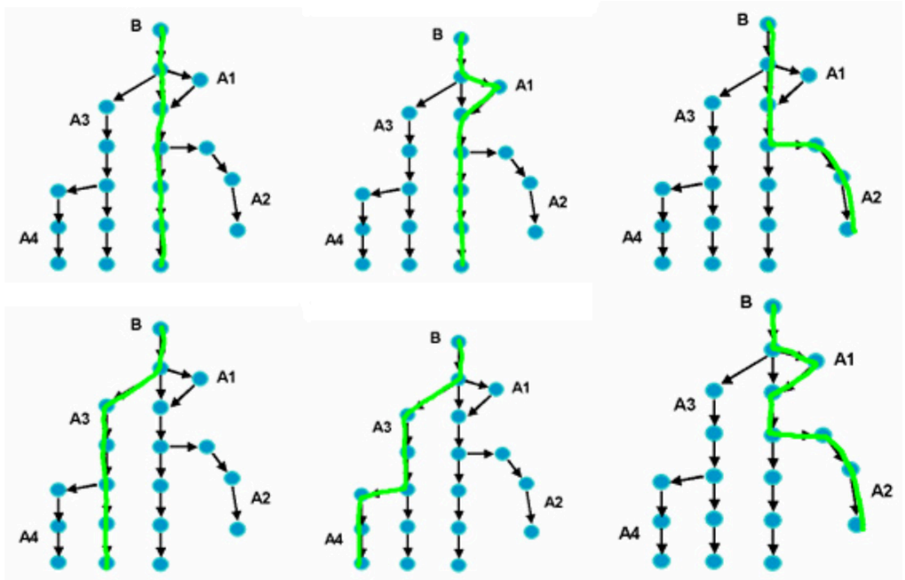
While a use case covers how users and system features work to reach goals, test cases verify if a single feature works correctly. Unlike use cases, test cases look at functionality in isolation.

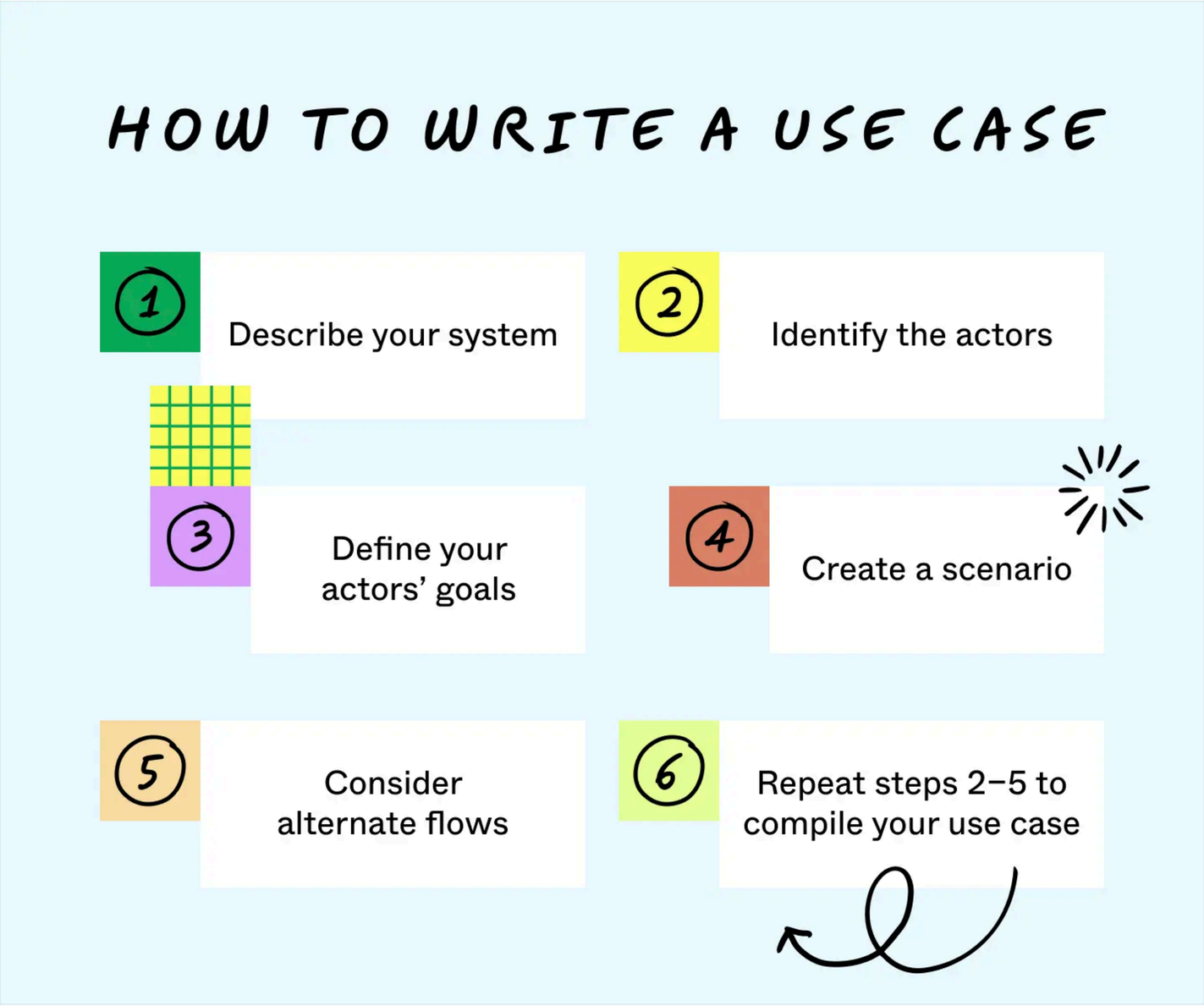
For example, a test case might involve validating login functionality on an email platform, ensuring users can log in on any browser at any time after creating their account.

# Use case towards validation test cases

Use Case Diagrams – for each use case examine possible scenarios, and choose a subset for **testing**. For example:






https://www.figma.com/resource-library/what-is-a-use-case/#

# 1. Describe your system

Start by describing your system, or the product or service you and your team will build.

Focus your description on what your system does for users. In a business use case, you can keep this background general and explain what it accomplishes. For a system use case, give an under-the-hood description of how your product functions.

Define your system by asking:

- What form does it take: product, service, or software?
- What features does it offer?
- What goals can you accomplish with it?
- How does it meet those goals?
- What can you learn about the system from other documents like project charters?

# 2. Identify the actors

Actors generally refer to users and customers but can apply to **any outside force that engages with your system**.

Your actor needs **well-defined behaviors** explaining how and why actors use your system.

Identify actors by asking:

- **What are they**: individuals, teams, hardware, or another system?
- Will **primary** and **secondary** actors share the same behavior?
- Will **stakeholders** take on the role of actors in your use case?

## 3. Define your actors' goals

- Use cases highlight the **outcome** actors want from a system.
- Remember to focus on **what your actors' want** over the system's capabilities to understand why users come to your system.
- In some cases, customers want to use systems for **more than one objective.**
- Listing each of these objectives creates a more **robust** use case.

## 4. Create a scenario

- In a use case, scenarios are the **sequence of actions** customers take when using a system and the **flow of effects** from that interaction.
- Your **basic flows** cover scenarios where a system works as intended. A user approaches the system, enters the right inputs, and your system helps them reach their goals.
- Start with these successful, basic flows to create a **baseline**.

## 5. Consider alternate flows

- After writing a successful scenario, write alternate flows that lead to different outcomes. Typically, alternate flows involve the misuse of a system that keeps actors from reaching their goals. However, you can also note internal errors that cause a system to break down or unintended ways systems can reach goals.
- Alternate flows show how different actors use a system and succeed or fail. They give a more nuanced view of everything your system can do to help you troubleshoot.

# 6. Repeat steps 2–5 to compile your use case

- With enough variation of actors, goals, and scenarios, you can show how your system functions. Compiling these flows together gives you a use case, which can improve development and inform other documents like project status reports.
- With simple systems, you can change a few elements to see every potential outcome. However complex systems may have too many elements to see each outcome. In cases like this, you can focus on testing the most common interactions. You can also design systems to prevent untested com

# Positive and Negative Scenarios



**One quite effective approach**:

try to **approximate** the requirements with positive and negative **scenarios**.

- Dear customer, please describe example usages of the desired system.

  Customer intuition: "If the system is not at all able to do this, then it's not what I want."
- Dear customer, please describe behaviour that the desired system must not show.

  Customer intuition: "If the system does this, then it's not what I want."
- From there on, refine and generalise:
  what about exceptional cases? what about corner-cases? etc.

https://swt.informatik.uni-freiburg.de/teaching/SS2017/swtvl/Resources/Slides/lecture-20170601-1-annot-fix-2up.pdf

# Positive and Negative Scenarios

CSC 4104

## *Example: Vending Machine*

- **Positive scenario**: Buy a Softdrink
  - (i) Insert one 1 euro coin.
  - (ii) Press the 'softdrink' button.
  - (iii) Get a softdrink.
- **Positive scenario**: Get Change
  - (i) Insert one 50 cent and one 1 euro coin.
  - (ii) Press the 'softdrink' button.
  - (iii) Get a softdrink.
  - (iv) Get 50 cent change.
- **Negative scenario**: A Drink for Free
  - (i) Insert one 1 euro coin.
  - (ii) Press the 'softdrink' button.
  - (iii) Do not insert any more money.
  - (iv) Get **two** softdrinks.

https://swt.informatik.uni-freiburg.de/teaching/SS2017/swtvl/Resources/Slides/lecture-20170601-1-annot-fix-2up.pdf

# User Stories (Beck, 1999)

*"A User Story is a **concise, written description** of a **piece of functionality** that will be **valuable to a user** (or owner) of the software."*

Per **user story**, use one **file card** with the user story, e.g. following the pattern:

*As a [role] I want [something] so that [benefit].*

and in addition:

- **unique identifier** (e.g. unique number),
- **priority** (from 1 (highest) to 10 (lowest)) **assigned by customer**,
- **effort**, **estimated by developers**,
- back side of file card: (acceptance) **test case(s)**, i.e., how to tell whether the user story has been realised.

**Proposed card layout** (front side):

| priority | unique identifier, name | estimation |
|---|---|---|
| *As a [role] I want [something] so that [benefit].* | | |
| risk | | real effort |



Beck, Kent. *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

# Natural Language Patterns

Natural language requirements can be (tried to be) written as an instance of the **pattern** "$\langle A \rangle$ $\langle B \rangle$ $\langle C \rangle$ $\langle D \rangle$ $\langle E \rangle$ $\langle F \rangle$." (German grammar) where

| | |
|---|---|
| $A$ | clarifies when and under what conditions the activity takes place |
| $B$ | is MUST (obligation), SHOULD (wish), or WILL (intention);<br>also: MUST NOT (forbidden) |
| $C$ | is either "the system" or the concrete name of a (sub-)system |
| $D$ | one of three possibilities:<br>• "does", description of a system activity,<br>• "offers", description of a function offered by the system to somebody,<br>• "is able if",<br>usage of a function offered by a third party, under certain conditions |
| $E$ | extensions, in particular an object |
| $F$ | the actual process word (what happens) |

(Rupp and die SOPHISTen, 2009)

**Example**:

After office hours ($= A$), the system ($= C$) should ($= B$) offer to the operator ($= D$) a backup ($= F$) of all new registrations to an external medium ($= E$).

# Use Case: Definition



**use case** – A **sequence of interactions** between an actor (or actors) and a system triggered by a specific actor, **which produces a result** for an actor. (Jacobson, 1992)

More precisely:

- A use case has **participants**: the **system** and at least one **actor**.
- **Actor**: an actor represents what interacts with the system.
  - An actor is a **role**, which a **user** or an **external system** may assume when interacting with the system under design.
  - Actors are not part of the system, thus they are **not described in detail**.
  - Actions of actors are **non-deterministic** (possibly constrained by domain model).
- A use case is triggered by a **stimulus** as input by the **main actor**.
- A use case is **goal oriented**, i.e. the main actor wants to reach a particular goal.
- A use case describes **all interactions** between the system and the participating actors that are needed to achieve the goal (or fail to achieve the goal for reasons).
- A use case **ends** when the desired goal is achieved, or when it is clear that the desired goal cannot be achieved.

# *Use Case Example: ATM Authentica*

| | |
|---|---|
| name | Authentication |
| goal | the client wants access to the ATM |
| pre-condition | the ATM is operational, the welcome screen is displayed, card and PIN of client are available |
| post-condition | client accepted,<br>services of ATM are offered |
| post-cond. in exceptional case | access denied, card returned or withheld, welcome screen displayed |
| actors | client (main actor), bank system |
| open questions | none |
| normal case | 1. client inserts card<br>2. ATM read card,<br>sends data to bank system<br>3. bank system checks validity<br>4. ATM shows PIN screen<br>5. client enters PIN<br>6. ATM reads PIN,<br>sends to bank system<br>7. bank system checks PIN<br>8. ATM accepts and shows main menu |
| exception case 2a | card not readable<br>2a.1 ATM displays "card not readable"<br>2a.2 ATM returns card<br>2a.3 ATM shows welcome screen |

| | |
|---|---|
| exc. case 2b | card readable, but not ATM card |
| exc. case 2c | no connection to bank system ✓ |
| exc. case 3a | card not valid or disabled ✓ |
| exc. case 5a | client cancels ✓ |
| exc. case 5b | client doesn't react within 5 s ✓ |
| exc. case 6a | no connection to bank system ✓ |
| exc. case 7a | first or second PIN wrong |
| exc. case 7b | third PIN wrong |

# Example: Use Case Diagram of the ATM Use Case

## Use Case Example: ATM Authentication

| | |
|---|---|
| name | Authentication |
| goal | the client wants access to the ATM |
| pre-condition | the ATM is operational, the welcome screen is displayed, card and PIN of client are available |
| post-condition | client accepted, services of ATM are offered |
| post-cond. in exceptional case | access denied, card returned or withheld, welcome screen displayed |
| actors | client (main actor), bank system |
| open questions | none |
| normal case | 1. client inserts card<br>2. ATM read card, sends data to bank system<br>3. bank system checks validity<br>4. ATM shows PIN screen<br>5. client enters PIN<br>6. ATM reads PIN, sends to bank system<br>7. bank system checks PIN<br>8. ATM accepts and shows main menu |
| exception case 2a | card not readable<br>2a.1 ATM displays "card not readable"<br>2a.2 ATM returns card<br>2a.3 ATM shows welcome screen |


http://commons.wikimedia.org/ CC-by-sa 4.0, Dirk Ingo Franke

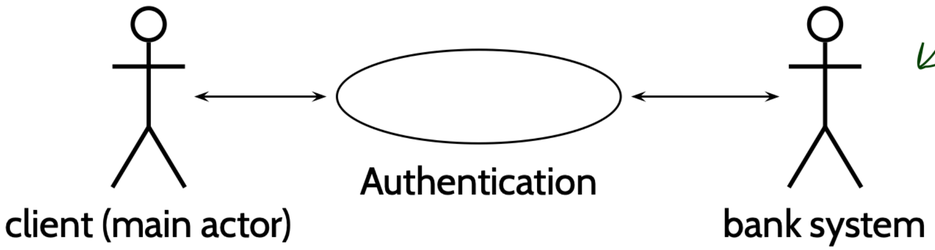
| | |
|---|---|
| exc. case 2b | card readable, but not ATM card |
| exc. case 2c | no connection to bank system |
| exc. case 3a | card not valid or disabled |
| exc. case 5a | client cancels |
| exc. case 5b | client doesn't react within 5 s |
| exc. case 6a | no connection to bank system |
| exc. case 7a | first or second PIN wrong |
| exc. case 7b | third PIN wrong |

(Ludewig and Lichter, 2013)

– 8 – 2017-06-01 – Suc –

14/27

abstraction



# Use Case - External to Internal

CSC 4104

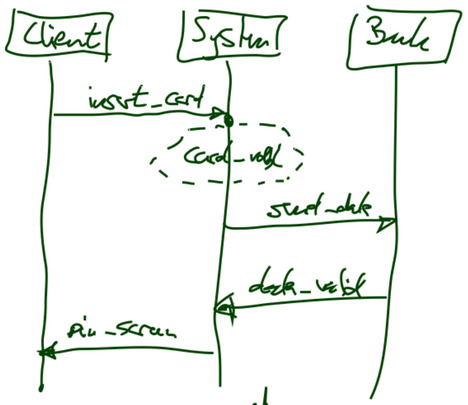
## *Use Case Example: ATM Authentication*

| | |
|---|---|
| name | Authentication |
| goal | the client wants access to the ATM |
| pre-condition | the ATM is operational, the welcome screen is displayed, card and PIN of client are available |
| post-condition | client accepted, services of ATM are offered |
| post-cond. in exceptional case | access denied, card returned or withheld, welcome screen displayed |
| actors | client (main actor), bank system |
| open questions | none |
| normal case | 1. client inserts card<br>2. ATM read card, sends data to bank system<br>3. bank system checks validity<br>4. ATM shows PIN screen<br>5. client enters PIN<br>6. ATM reads PIN, sends to bank system<br>7. bank system checks PIN<br>8. ATM accepts and shows main menu |
| exception case 2a | card not readable<br>2a.1 ATM displays "card not readable"<br>2a.2 ATM returns card<br>2a.3 ATM shows welcome screen |

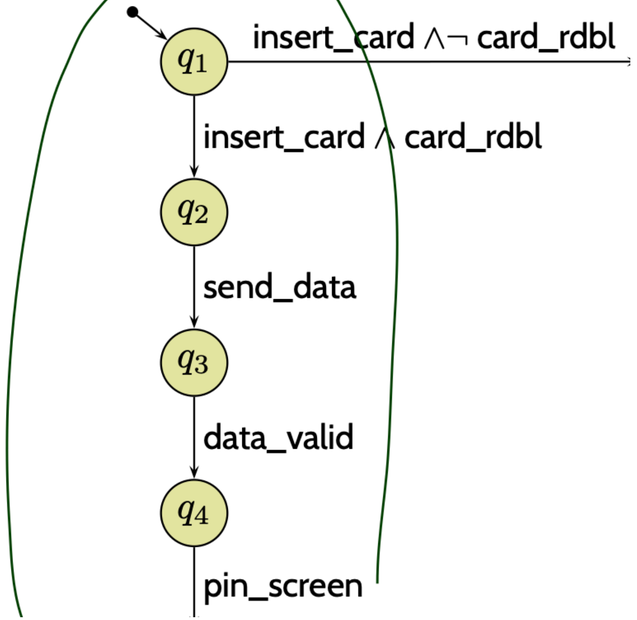

| | |
|---|---|
| exc. case 2b | card readable, but not ATM card |
| exc. case 2c | no connection to bank system |
| exc. case 3a | card not valid or disabled |
| exc. case 5a | client cancels |
| exc. case 5b | client doesn't react within 5 s |
| exc. case 6a | no connection to bank system |
| exc. case 7a | first or second PIN wrong |
| exc. case 7b | third PIN wrong |

(Ludewig and Lichter, 2013)

14/27




## (2.) Finite Automaton:



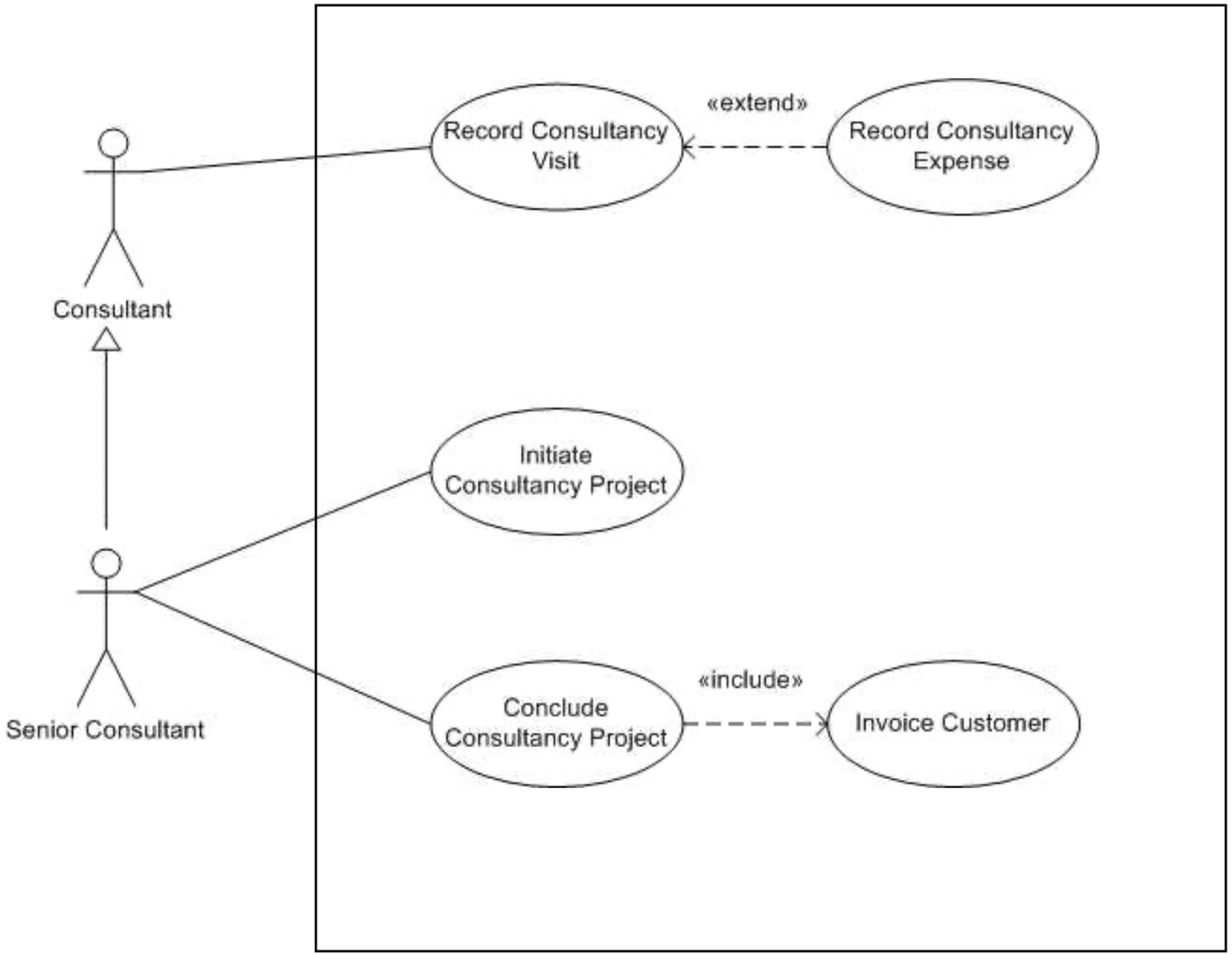

## (1.) Observables:

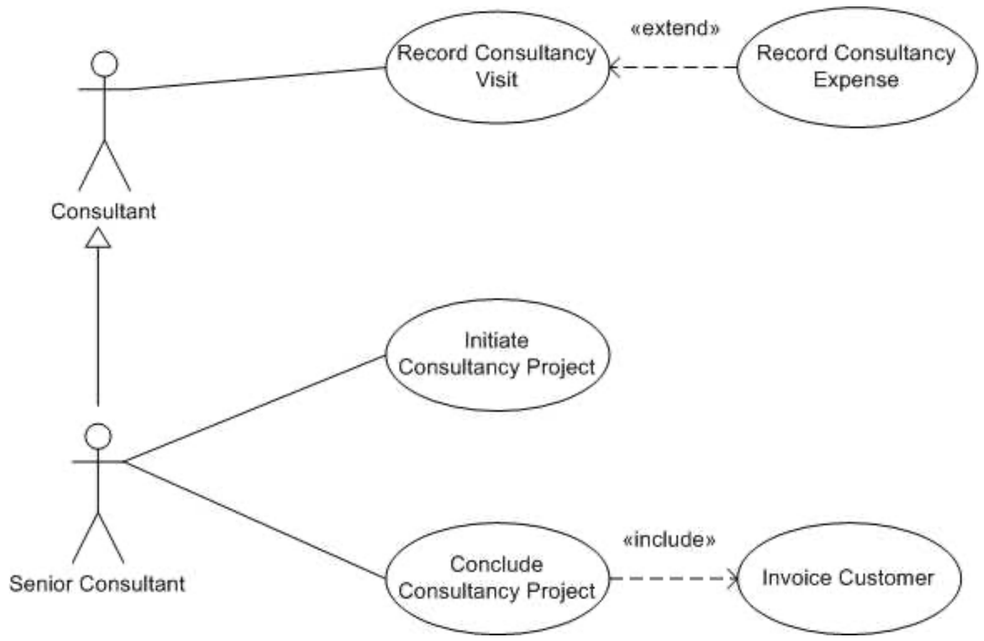
- event **insert_card**
- condition **card_rdbl**
- event **send_data**
- event **data_valid**
- event **pin_screen**

| Element | Definition |
|---|---|
| Use case | Set of operations performed by/in system that produces measurable result for an actor |
| Actor | Set of roles that users (can be a system) play |
| System | Boundary between a software/hardware/manual system and other actors or systems |
| Association | Participation of an actor in a use case |
| Generalization | Relationship between general and more specific actor or use case. Arrow points to general use case or actor |
| Include | Variation on base use case |
| Extend | Some modelers (not us) make the following distinction:<br>Include is used when a common use case is inserted in two or more base cases: e.g., “login” used both by “make reservation” and “cancel reservation”<br>Extend is used when a variation is inserted in only one base case: e.g., “make multiple reservations” extends “make reservation” |

# **UML** Use case diagram example








In this diagram we used a **generalization** relationship between two actors as Senior Consultant can do everything a consultant can do so basically senior consultant is a consultant with some extra responsibilities so in all such cases we can use generalization between actors.

We can use **extend** relation between "Recording Consultancy Visit" and "Recording Consultancy Expense" because "Recording Consultancy Expense" use case is a conditional use case which is only called if consultancy visit is a paid visit.

We used an **include** relation between "Conclude Consultancy Project" and "Invoice Customer" as "Invoice Customer" use case must be called when a project concludes, in any such condition when we have a use case that must be called after a use case then there will be an include relationship.

# How to use include and extend wisely

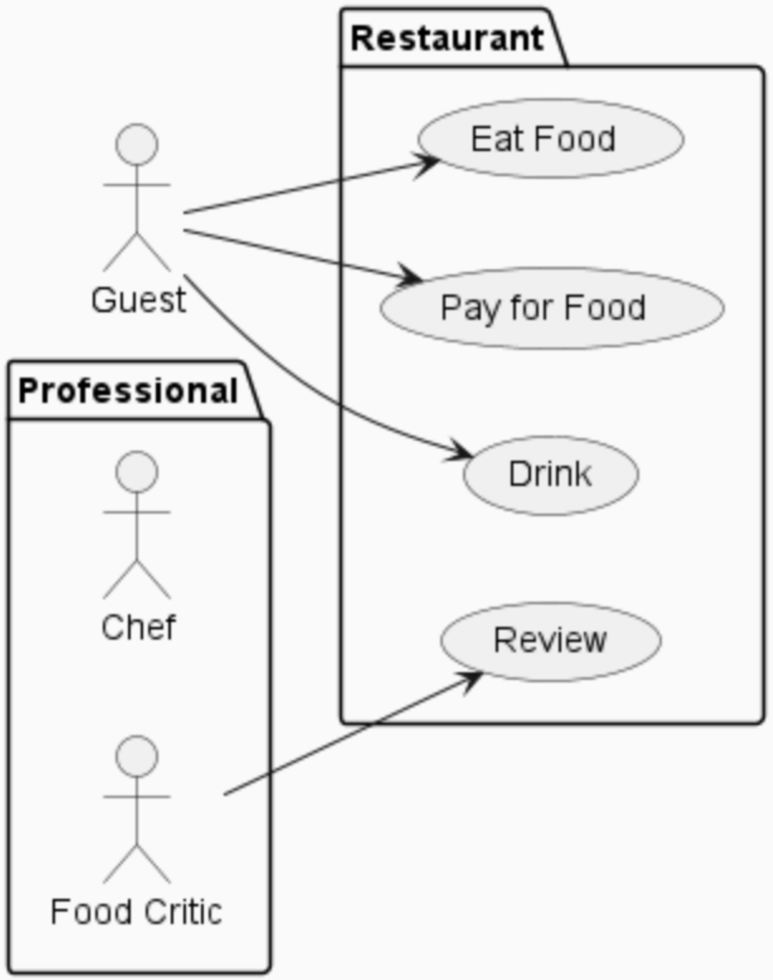
To ensure the effective use of include and extend relationships in your use case diagram, it’s important to follow certain guidelines.

Include relationships should be used for common and mandatory behaviors that are relevant to the main goal of your use cases.

Extend relationships should be used for optional and conditional behaviors that are not essential to the main goal of your use cases.

# Use Case Diagrams With PlantUML https://plantuml.com/use-case-diagram

```
@startuml
left to right direction
actor Guest as g
package Professional {
  actor Chef as c
  actor "Food Critic" as fc
}
package Restaurant {
  usecase "Eat Food" as UC1
  usecase "Pay for Food" as UC2
  usecase "Drink" as UC3
  usecase "Review" as UC4
}
fc --> UC4
g --> UC1
g --> UC2
g --> UC3
@enduml
```



# Overview of a User-story-driven Process

• A software development process driven by user stories feels very different than traditional life cycles; f instance, customers are included throughout the process (they do not disappear on you!)

• to get a project started, a story writing workshop is held to brainstorm what features are valuable to th customer for an initial release

• developers will assign initial estimates to each story using “points”

• customers and developers set an iteration length (e.g. 2 weeks)

• developers then determine their velocity (how much work they can get done in a single iteration)

• customers assign priorities to the stories

• iterations are formed by grouping stories by velocity based on their priorities and estimates

https://home.cs.colorado.edu/~kena/classes/5828/f16/lectures/08-userstoriesiterationplan.pdf

# Estimating User Stories

CSC 4104

• Developers need to assign “points” to a story to indicate how long it will take to implement

- Our user/customer assigns priorities to stories, not estimates
- There are a number of desirable properties for this approach
  - it allows us to change our minds about an estimate when new info arrives
  - works for both epic stories as well as smaller stories

• doesn’t take a lot of time; we want to spend our time developing

• provides useful information about our progress and work remaining

• is tolerant of imprecision in estimates • can be used to plan releases

# Assigning Priorities

- One prioritization scheme that may be better than the typical "low/medium/high" approach
    - Must have
    - Should have
    - Could have
    - Won't have (for this release)
- This approach divides stories into clear buckets that can then be used to assign st to iterations within the release
- If a customer can't assign a priority to a user story, this (typically) indicates that the story needs to be split until clear priorities can be assigned

# Avoid or embrace risk?

- The issue here is what approach should agile projects take • tackle risky stories first • or go after “low hanging fruit”
- Agile life cycles like to go after low-hanging fruit
  • ***high-value functionality that is straightforward to implement***
- This allows time for more information to be gathered about high-risk stories • and this additional information may reduce the risk associated with them
- I think you need to balance this with the common issue of “problem avoidance”; make sure you’re clear on what the risks are => such information may produce action items that can reduce the risk and make it feasible

# TODO - PBL - User stories for the “Feuille de présence dématérialisée” (FdPD)

- Write 8-16 user stories for the FdPD case study application
- Draw UML Use case diagrams that incorporate the uses cases of the stories
- Assign priorities
- Estimate development ‘time’

# User stories for the “Feuille de présence dématérialisée” (FdPD) - some useful examples

CSC 4104

https://www.inflectra.com/Ideas/Topic/Use-Cases.aspx

Example 1

| | |
|---|---|
| Use Case #1 | Quiz Instant Feedback |
| Description | An educational technology company wants to develop a feature that allows students to take quizzes and receive instant feedback. |
| Actors | Students |
| Goals | Take a quiz, view quiz results |
| Stakeholders | Educational technology company, students, educators, school administration, investors |
| Pre-conditions | Student must be logged in, student must have access to the quiz |
| Post-conditions | Student can take the quiz and receive instant feedback on their performance |
| Basic flow | Student logs into the educational technology platform and selects the option to take a quizSystem presents the student with the quiz questionsStudent answers the questions and submits the quizSystem evaluates the student's answers and provides instant feedback on their performanceStudent views the feedback and can review their answers if they desireSystem records the quiz results and makes them available for the student, educator, and administration to view |
| Alternate path | Student logs into the educational technology platform and selects the option to view previous quiz resultsSystem presents the student with a list of previous quizzes they have taken, along with their resultsStudent selects a previous quiz to viewSystem displays the student's answers and the correct answers for each questionStudent reviews their answers and receives feedback on their performance |