Use Case Diagrams

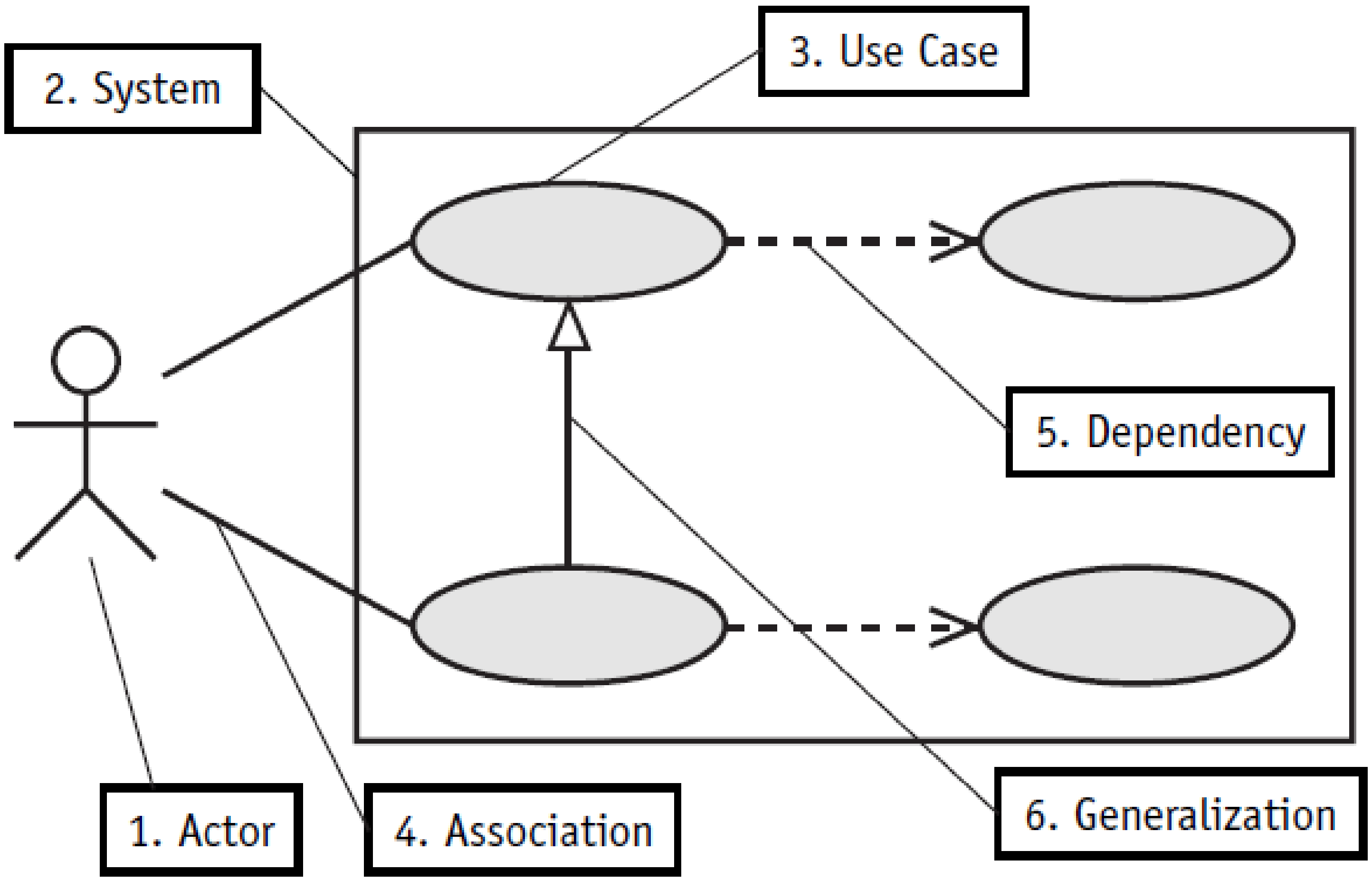
Citra N., MT

Definition

- A diagram use case is typical interaction between a user and the system under development. It is used to capture some functionality to be provided by the software system.
- A use case is a usage of the system that provides an observable and (usually) meaningful result. The use-case documentation (diagrams and/or text) should delineate the series of steps that take place during the interaction and include different ways that this interaction could play out.

- use cases are used to document our understanding of the way a business operates – business requirements modeling – and to specify what our new software system should be able to do – system requirements modeling.
- The business use cases can use an informal, descriptive style: they describe, for the benefit of nonexperts, something that already exists. The system use cases, on the other hand, will be more prescriptive: they specify, mainly for the benefit of software developers, exactly what functionality needs to be implemented.

Notation


2. System
3. Use Case
5. Dependency
6. Generalization
1. Actor
4. Association

- System : sets the boundary of the system in relation to the actors who use it (outside the system) and the features it must provide (inside the system)
- Actor : A role played by a person, system or device that has a stake in te succesful operation of the system.
- Use case : Identifies a key feature of the system. Without these features, the system will not fulfill the user/actor requirements, each use case expresses a goual the system must achieve.
- Association : Identifies an interaction between actors and use cases, each association becomes a dialog that must be explained in a use case narrative. Each narrative in turn provides a set of scenarios that function as test cases when evaluating the analysis, design and implementation of the use case

Actors

- In this step, you identify the IT system’s users, or actors. An actor specifies a role played by a user or any other system that interacts with the subject
- An actor is a type of user or an external system that interacts with the system under design
- **External agent/external entity**: Equivalent terms used in structured analysis.
- **Stakeholder**: A term more inclusive than actor as it includes anyone who the project will affect even if they do not have direct contact with it.

Finding Actors

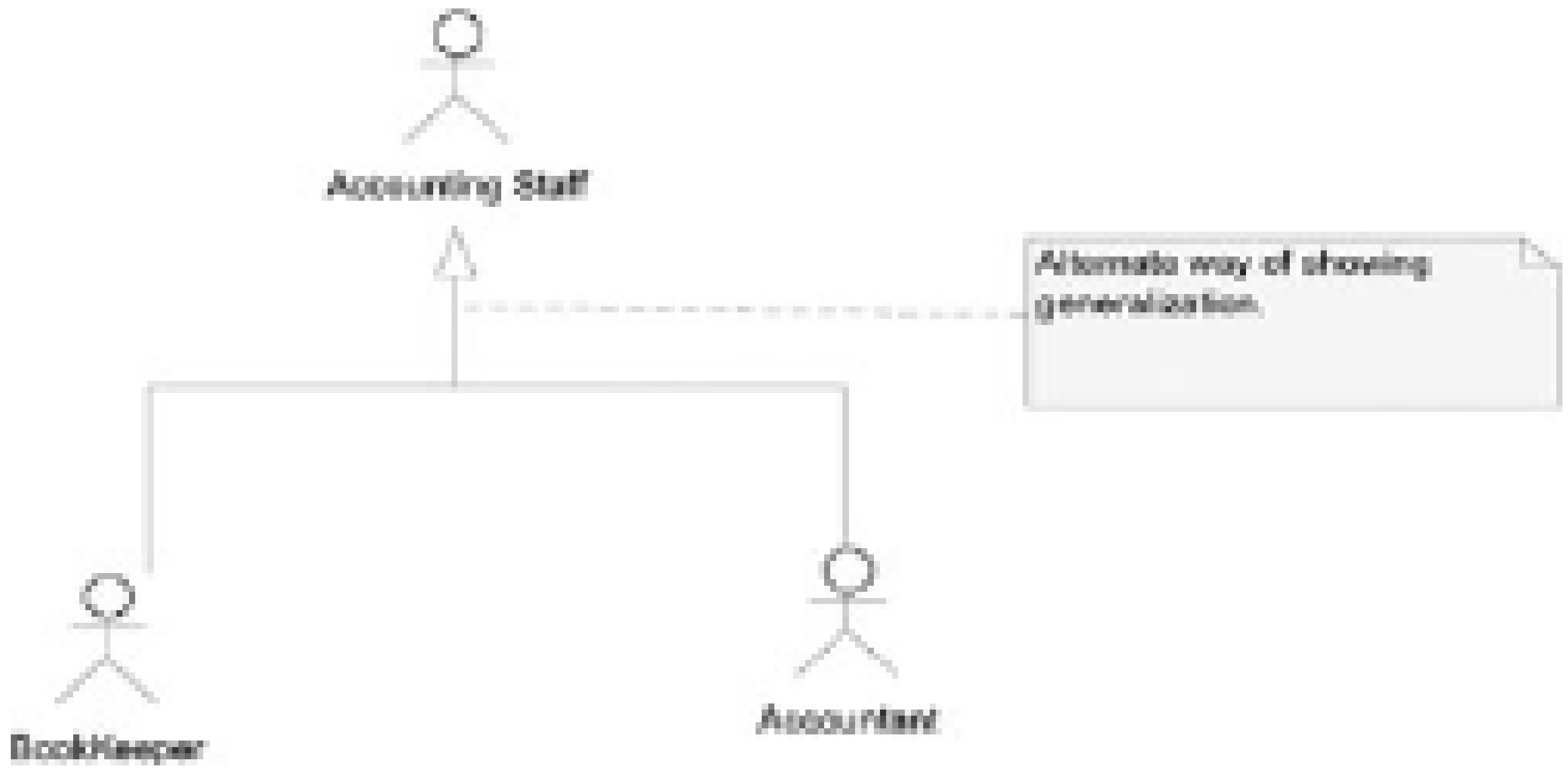
- To find actors, go through your list of business actors and workers, eliminating any who don’t interact with the IT system. Then add any external systems and human users who are required because of the technology.

The Role Map

- A role map is a diagram used to standardize the treatment of users and external systems throughout the project. A role map is a restricted form of a use-case diagram. Whereas the use-case diagram shows actors and their associations with use cases, the role map shows only actors.
- Place icons for each of the actors you’ve identified in the role map. The role map then becomes the central diagram team members go back to whenever they want to know how to depict a user in the model. You can also use the role map to show the ways in which user roles overlap.

Modeling Actors with Overlapping Roles

- You document actors with overlapping roles by drawing a generalization relationship between actors. Any time the phrase “a kind of” comes up in the discussion of actors, think about using the generalization relationship.
- When two actors have some overlap in their roles, but each actor can do things with the system that the other can’t, model the actors as specialized actors and invent an abstract generalized actor to represent the overlap. The term generalized implies that the specialized actors inherit something from the generalized actor. In this case, the specialized actors inherit the ability to do all the things that the generalized actor can do.

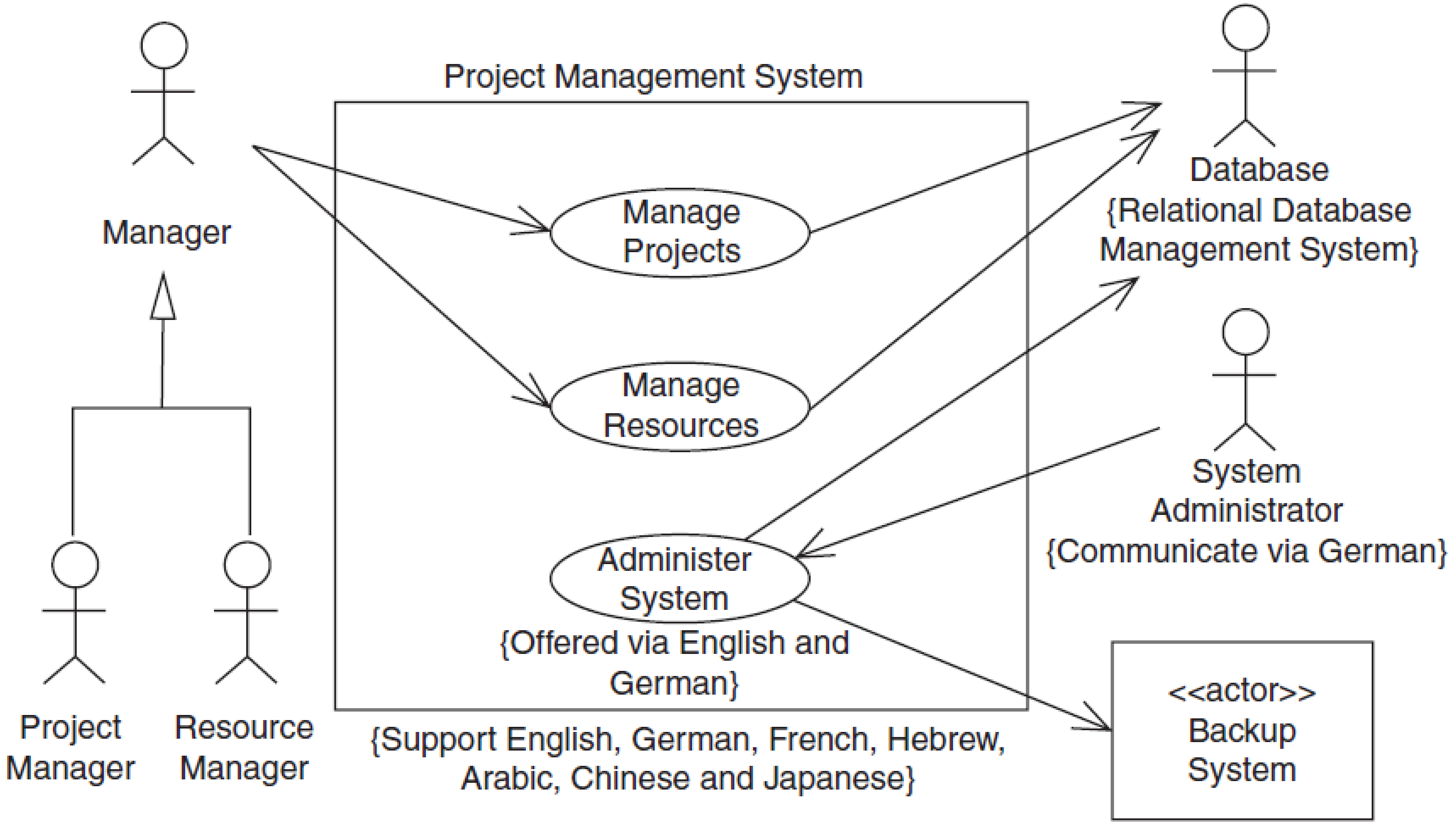

Accounting Staff
Alternate way of showing generalization.
BookKeeper
Accountant

System Use-Case Diagram

- If your project supports only one business use case, you may proceed directly to the following step, identify system use cases. But if it supports a number of business use cases, consider creating system use-case packages. A system use-case package is a collection of system use cases and the diagrams that describe them.
- Group system use cases by the main actor who uses them. For example, group together into one package all the system use cases used by general administration.

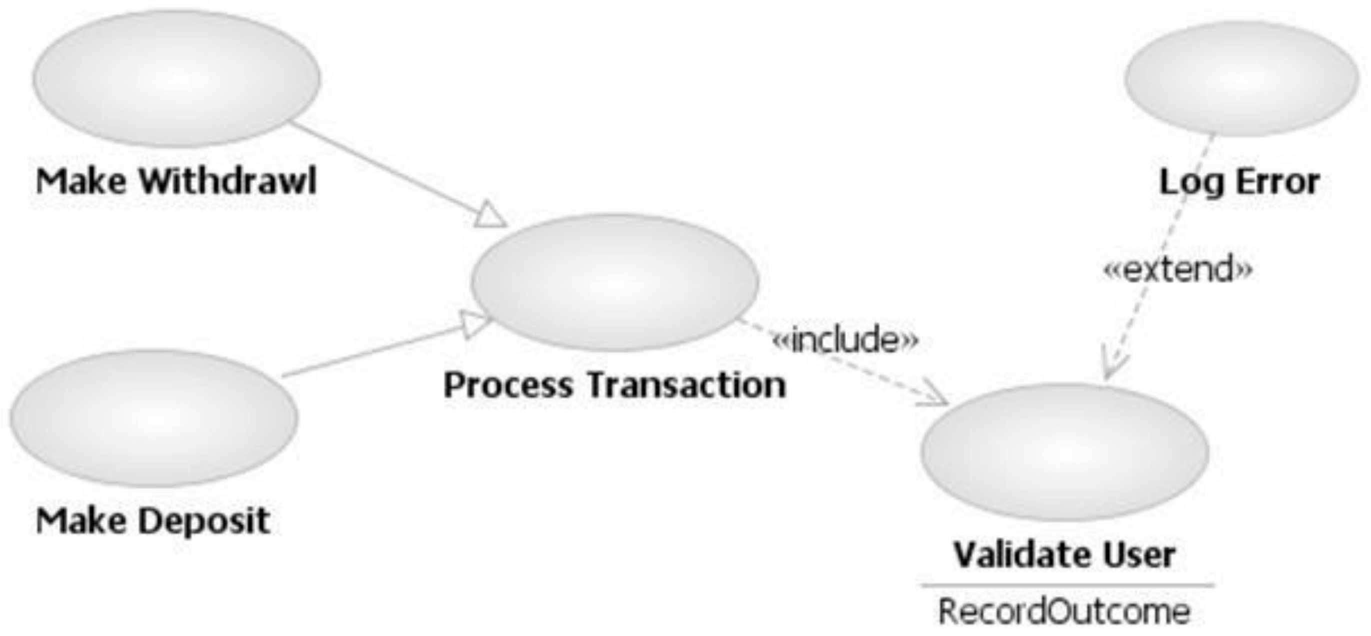
System Use Cases

- The system use-case diagram does not show sequencing; you can't tell from the diagram the order in which the system use cases should be used or the sequence of activities within each use case.
- Primary actor: An actor who initiates a use-case interaction (indicated as an actor at the tail end of an arrow pointing to a use case).
- Secondary actor: An actor that the system initiates an interaction with after the use case has started (indicated as an actor at the tip of an arrow pointing from a use case).
- System use case: A user task (indicated as an oval).


Project Management System
Manager
Manage Projects
Manage Resources
Administer System
{Offered via English and German}
Database
{Relational Database Management System}
System Administrator
{Communicate via German}
<<actor>>
Backup System
Project Manager
Resource Manager
{Support English, German, French, Hebrew, Arabic, Chinese and Japanese}

Use-case Relationship

- Specializes: Just like actors, use cases can inherit from each other. In order to avoid all sorts of complexity relating to the redefinition of steps and the addition of extra ones.
- Include: Extract the common requirements into a “mini use case.” The main use cases are said to include this mini use case. This approach is useful whenever a set of steps appears in more than one system use case.
- Extend: You leave one main system use case intact and create a new use case that extends the original one. The extending use case contains only the requirements that differ from the original. This approach is useful, for example, for enhanced versions of earlier software releases.
- Generalization: You create a new generalized use case that contains general rules. Other use cases are created as “specializations”; they contain the non-generic requirements. This approach is useful when a set of system use cases represents variations on a theme


Make Withdrawl
Log Error
«extend»
Process Transaction
«include»
Make Deposit
Validate User
RecordOutcome

Task 1

- Outpatient Information System at PUSKESMAS XXX

1. Patients give their data to the registration, then examined whether patient registered or not. If patients have not been registered yet, then patient submits the data - the data to the registration, and then made cards and card ambulatory outpatient. Data - the data is returned to the patient along with the patient's outpatient card, while the card is stored at the outpatient registration.
2. But if the patient is already enrolled, the patient handed medical card or health insurance card along ASKESKIN if there is, then based outpatient card, officers look for medical record - her and recorded the data - its data in patients and in the ticket book. Book kept at the patient registration, tickets and medical record being submitted to the examination. From the book made the list of patient visits patients and submitted to the head of the health center reports.

Use case narrative

Elements of Use Case Narrative

- Describing a use case requires that you frame the context of the use case and describe the communication between the use case and teuser, which could be an actor or another use case.

The Use Case Text (Scenario)

- Scenario : A specific sequence of actions that illustrates behaviors. A scenario may be used to illustrate an interaction or the execution of a use-case instance.
- A scenario is one path through a use case—one way that it might play out.
- When writing scenario, it’s important that we specify the function of the system.

- The use case number and title.
- Whether the use case is abstract.
- relationships to other use cases.
- Any preconditions (conditions that must be satisfied before the use case is carried out).
- The steps themselves (where we can assume that the preconditions have been met).
- Any postconditions (conditions that are guaranteed after successful completion of the use case).
- Any abnormal paths and what to do in each case (although the paths are abnormal, we include them if it’s important for us to specify the system’s reaction).
- Any nonfunctional requirements that relate to this use case.

The Use-Case Description

- The underlying principle of this template is to describe workflow using a simple narrative style that avoids complex logic. The trick to keeping things simple is to handle variations in a separate area of the document rather than in one all-encompassing section. First, you document a normal, typical interaction in a section called “Basic Flow.” Next, you describe alternative success scenarios in an “Alternate Flows” section. Finally, you describe error handling in an “Exceptional Flows” section.

Use-Case Description Template

1. Use Case: The use-case name as it appears on system use-case diagrams
   Perspective: Business use case/system use case
   Type: Base use case/extending/included/generalized/specialized

   1.1 Brief Description: Describe the use case in approximately one paragraph.
   1.2 Business Goals and Benefits: Briefly describe the business rationale for the use case.
   1.3 Actors
   1.3.1 Primary Actors: Identify the users or systems that initiate the use case.
   1.3.2 Secondary Actors: List the users or systems that receive messages from the use case. Include users who receive reports or online messages.
   1.3.3 Off-Stage Stakeholders: Identify non-participating stakeholders who have interests in this use case.

1.4 Rules of Precedence

1.4.1 Triggers: Describe the event or condition that "kick-starts" the use case, such as Call received; inventory low. If the trigger is time-driven, describe the temporal condition, such as End-of-month.

1.4.2 Pre-conditions: List conditions that must be true before the use case begins. If a condition forces the use case to occur whenever it becomes true, do not list it here; list it as a trigger.

1.5 Post-conditions

1.5.1 Post-conditions on Success: Describe the status of the system after the use case ends successfully. Any condition listed here is guaranteed to be true on successful completion.

1.5.2 Post-conditions on Failure: Describe the status of the system after the use case ends in failure. Any condition listed here is guaranteed to be true when the use case fails as described in the exception flows.

1.6 Extension Points: Name and describe points at which extending use cases may extend this use case. Example of extension point declaration: "Preferred Customer: 2.5–2.9."

2. Flow of Events

Basic Flow: Insert basic flow steps. Numbers begin with 2.1.

Alternate Flows

2.Xa Insert the alternate flow name. The alternate flow name should describe the condition that triggers the alternate flow. “2.X” is the step number in the basic flow where the interruption occurs. Describe the steps in paragraph or point form.

Exception Flows

2.Xa Insert the exception flow name. The exception flow name should describe the condition that triggers the exception flow. An exception flow is one that causes the use case to end in failure and for which “post-conditions on failure” apply. “2.X” is the step number in the basic flow where the interruption occurs. Describe the steps in paragraph or point form.

More Explanations …

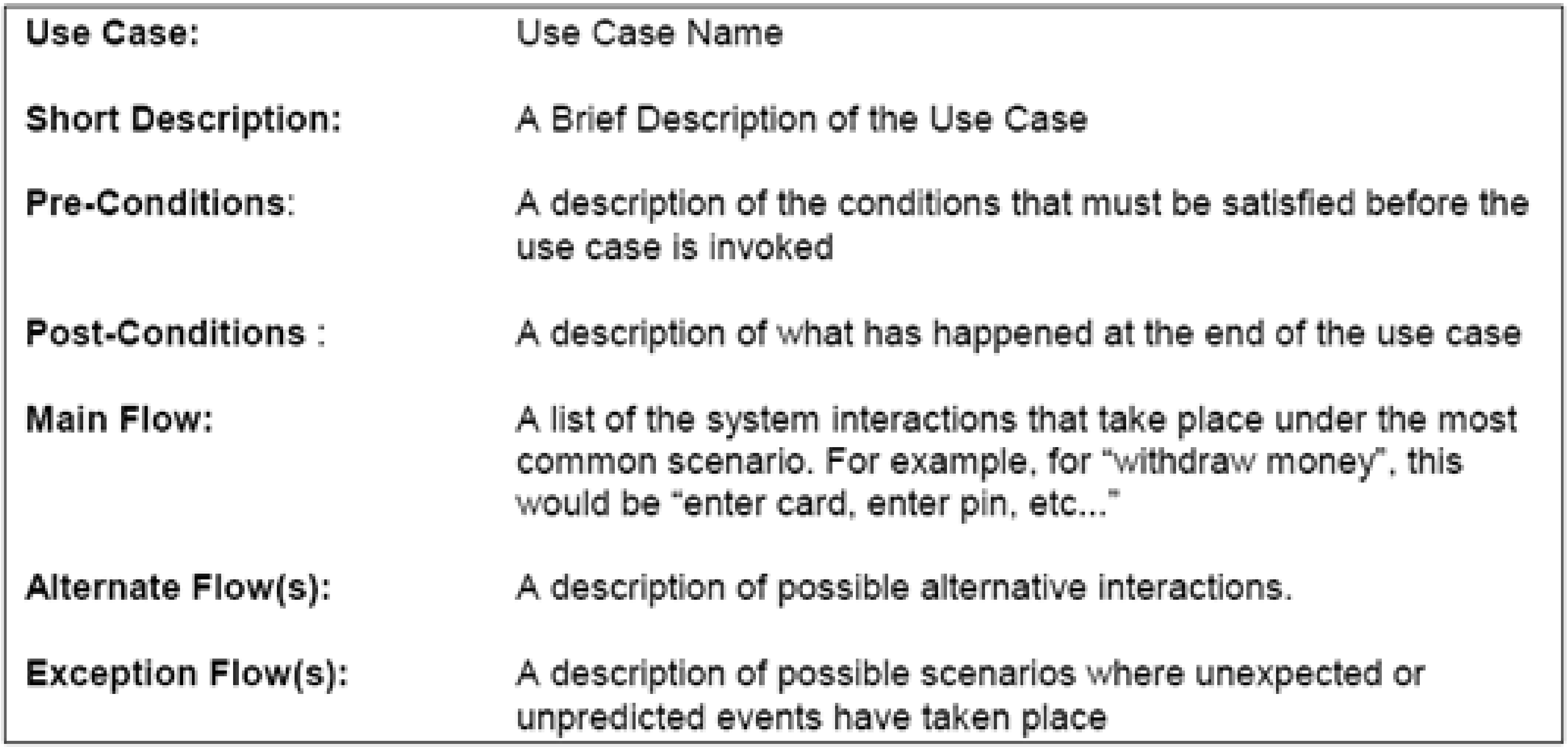
- Documenting the Basic Flow

  The basic flow describes the most common way that the use case plays out successfully. (Some people call it the "happy scenario.") It reads as a straightforward narrative: "The user does…; the system does…." As a rule of thumb, the basic flow should not list any conditions, since subsequent sections handle all errors and alternatives. To keep documentation consistent, employ a style guideline throughout your company for writing use-case requirements.
- Documenting Alternate Flows

  Document each scenario not covered in the basic flow as an alternate flow or as an exception flow. An alternate flow is a variation that does not lead to the abandonment of the user goal; an exception flow involves a non-recoverable error. If your team has trouble deciding whether to list a scenario in the "Alternate Flow" or "Exception Flow" section, merge the two sections into one and list both types of flows there.
- An alternate flow is a scenario other than the basic flow that leads to success. An alternate flowmay deal with a user error as long as it is recoverable. Non-recoverable errors are handled as exception flows.

- Documenting Exception Flows

  List each error condition that leads to the abandonment of the user goal in the "Exception Flows" section. Typical exception flows include cancellation of a transaction by the user and system errors that force a transaction to be canceled. Documentation rules are the same as for the alternate flows except that there is often no convergence point, since the goal is abandoned. In that case, the last line of the flow should read, "The use case ends in failure."



Use Case:	Use Case Name
Short Description:	A Brief Description of the Use Case
Pre-Conditions:	A description of the conditions that must be satisfied before the use case is invoked
Post-Conditions :	A description of what has happened at the end of the use case
Main Flow:	A list of the system interactions that take place under the most common scenario. For example, for "withdraw money", this would be "enter card, enter pin, etc..."
Alternate Flow(s):	A description of possible alternative interactions.
Exception Flow(s):	A description of possible scenarios where unexpected or unpredicted events have taken place

Use Case Name	Withdraw Name
Primary Actor	Customer
Supporting Actor(s)	Bank Accounting System
Summary	Custumer withdraws cash from the ATM system by inserting his or her card, entering the correcting PIN, selecting an account, and entering an amount. The ATM system validates the card, PIN, account and amount with the Bank Accounting System.
Pre - Conditions	1. ATM has money and supplies. 2. Bank accounting system is working.
Normal Flow of Events	1. User inserts ATM card. 2. ATM reads and validates bank ID and account number with bank account system. 3. User enters PIN number. 4. ATM validates PIN with bank accounting system. 5. User selects account. 6. User enters amount to withdraw. 7. ATM validates amount with bank accounting system. 8. ATM dispenses cash and receipt. 9. ATM logs transactions. 10. User takes card, cash and receipt.

Extensions	**1. Non-ATM card entered.ATM card inserted incorrectly.** **2. ATM card inserted incorrectly.** **3. Bank ID or account invalid.** **4. Card is from ineligible bank.** **5. Card is stolen.** **6. Customer does not enter PIN in time.** **7. PIN is invalid.** **5.1. Account is invalid.** **6.1. Amount is invalid or over maximum allowed.** **7.1. Insufficient funds in account.**
Post – Conditions	1. User’s account balance is adjusted. 2. ATM’s money inventory is adjusted. 3. ATM’s supply inventory is adjusted.