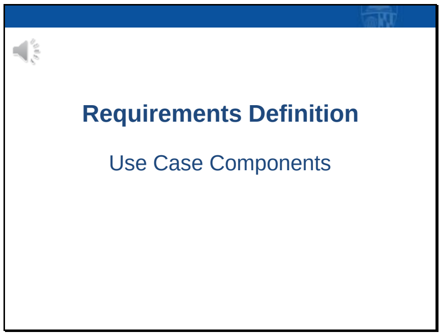
1

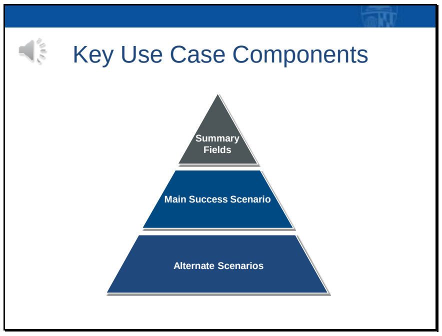


Requirements Definition

Use Case Components

In this lecture I’ll discuss the key components of a use case and describe how to complete the basic summary information for a use case.

2



Key Use Case Components

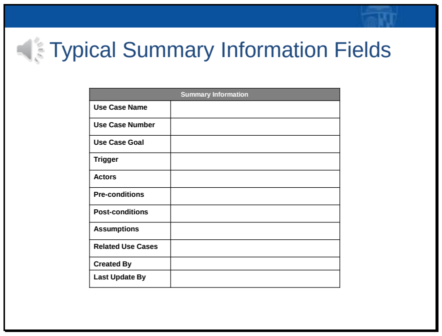
There are three primary components to a use case: a set of what I refer to as summary fields, a main success scenario, and typically one or more alternate scenarios.

The summary fields contain information like the use case name, use case participants, use case goal, assumptions, and so forth.

The main success scenario contains the typical interactions between users, other systems, and the system under development.

Alternate scenarios consist of interactions that describe alternate ways to achieve the goal of a use case, as well as interactions required to handle errors.

3

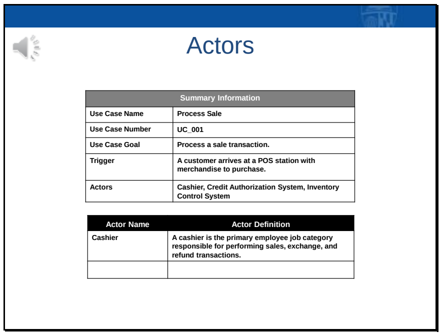


Typical Summary Information Fields

Summary Information	
Use Case Name	
Use Case Number	
Use Case Goal	
Trigger	
Actors	
Pre-conditions	
Post-conditions	
Assumptions	
Related Use Cases	
Created By	
Last Update By	

Here's a template that contains some common use case summary fields. Some of them are pretty self-evident. I'm going to pick a few of them and discuss them in more detail.

4



Actors

Summary Information	
Use Case Name	Process Sale
Use Case Number	UC_001
Use Case Goal	Process a sale transaction.
Trigger	A customer arrives at a POS station with merchandise to purchase.
Actors	Cashier, Credit Authorization System, Inventory Control System

Actor Name	Actor Definition
Cashier	A cashier is the primary employee job category responsible for performing sales, exchange, and refund transactions.

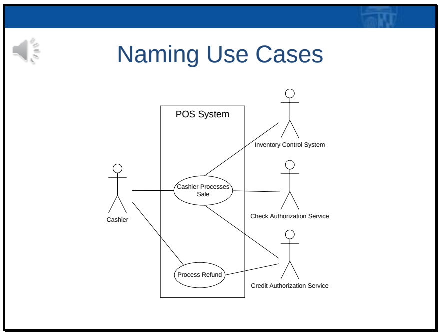
I'm going to start with actors. An actor is someone or some thing that interacts with the system we are gathering requirements for. Actors can be users, other systems, or time. Time is a kind of unusual actor, and one that is sometimes misunderstood. As an example, there may be a requirement that a report be generated at midnight each day that summarizes all the sale transactions for that day. Time would be an actor in such a use case.

The actors that participate in a use case are typcially documented in an actors field...and an actor glossary is created that provides brief descriptions of each actor.

Actor names should be meaningful in the business domain. Actor names often correspond to roles that get played by users of the system or job positions in an organization. Actor names shouldn't be too abstract. The reader of a use case should be able to recognize most actors based upon the role they play in the business domain.

There are two important rules when it comes to actors: actors must be external to the system we are gathering requirements for, and, a use case must have at least one actor.

5



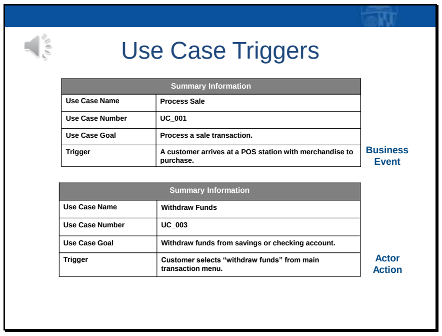
Naming Use Cases

Every use case must have a name...and the name should give us some idea about the goal of the use case...as you can see from these examples.

There are two basic forms that use case names take on: Actor-action-subject...as in Cashier Processes Sale, and action-subject...as in process refund. In practice, we pick one form and use it consistently.

By the way, this is an example of a simple use case diagram called a context diagram. Actors are shown as stick figures, use cases are shown as ovals, and the lines connecting the actors to the use cases indicate which actors participate in which use cases. The lines are technically called links. The rectangle is used to denote the scope of the system we are gathering requirements for.

6



Use Case Triggers

Summary Information	
Use Case Name	Process Sale
Use Case Number	UC_001
Use Case Goal	Process a sale transaction.
Trigger	A customer arrives at a POS station with merchandise to purchase.

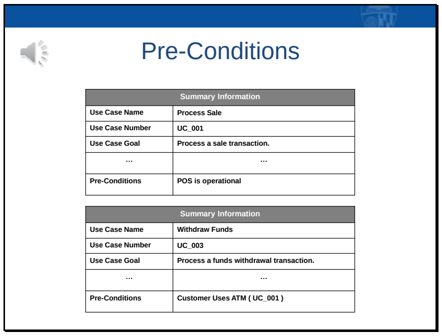
Business Event

Summary Information	
Use Case Name	Withdraw Funds
Use Case Number	UC_003
Use Case Goal	Withdraw funds from savings or checking account.
Trigger	Customer selects “withdraw funds” from main transaction menu.

Actor Action

Another common use case information field is a trigger field. A trigger is an event that causes a use case to start. It may be a business event...for example, in the process sale use case a customer may arrive at a point of sale station with merchandise to purchase...which causes a cashier to start a new sale transaction, or it may be an action that an actor takes that causes a new use case to start. As an example, in a bank atm system, a customer is presented with a menu of transaction options. Selecting the withdraw funds option causes the withdraw funds use case to start.

7



Pre-Conditions

Summary Information	
Use Case Name	Process Sale
Use Case Number	UC_001
Use Case Goal	Process a sale transaction.
…	…
Pre-Conditions	POS is operational

Summary Information	
Use Case Name	Withdraw Funds
Use Case Number	UC_003
Use Case Goal	Process a funds withdrawal transaction.
…	…
Pre-Conditions	Customer Uses ATM (UC_001)

I'm pretty sure that every use case template I've seen has had a pre-conditions field. And, many project teams I've worked with didn't quite understand what pre-conditions are all about.

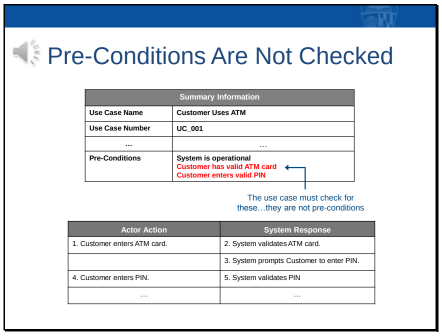
A pre-condition describes the state of the system that is required before a use case can attempt to start.

There are a few important things to remember about pre-conditions: First, anything a use case must check for is not a pre-condition. Second, use cases can be pre-conditions for other use cases, and third, it is not mandatory that a use case have pre-conditions.

Pre-conditions help give context to the reader of a use case, and also to developers who need to implement the software components required to support a use case.

Sometimes, pre-conditions are trivial…like the system must be operational. In those cases, it's okay to say there are no pre-conditions…because it's quite obvious. When one use case branches to another use case, the use case branched from becomes a pre-condition for the use case branched to.

8



Pre-Conditions Are Not Checked

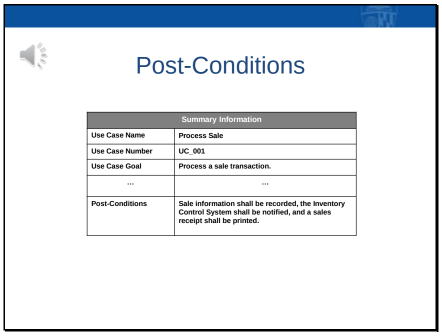
Summary Information	
Use Case Name	Customer Uses ATM
Use Case Number	UC_001
…	…
Pre-Conditions	System is operational Customer has valid ATM card Customer enters valid PIN

The use case must check for these…they are not pre-conditions

Actor Action	System Response
1. Customer enters ATM card.	2. System validates ATM card.
	3. System prompts Customer to enter PIN.
4. Customer enters PIN.	5. System validates PIN
…	…

It's extremely important to understand that use cases don't check whether pre-conditions are true…so…anything a use case must check for can't be one of its pre-conditions. Take a look at this example for a bank ATM system. The use case name is Customer Uses ATM, and it describes the interactions involved in an ATM session. A customer logs in by entering an ATM card and a valid PIN, the system then presents the customer with a menu of transaction types, the customer chooses a transaction type, and the interactions continue until the customer ends the ATM session. The first five steps of the use case are shown here. Note that the system must validate the card and the PIN…so it is incorrect to specify these as pre-conditions.

9



Post-Conditions

Summary Information	
Use Case Name	Process Sale
Use Case Number	UC_001
Use Case Goal	Process a sale transaction.
…	…
Post-Conditions	Sale information shall be recorded, the Inventory Control System shall be notified, and a sales receipt shall be printed.

Post-conditions are another very common field. A post-condition describes the state of the system after a use case has taken place successfully. Successfully is a key word here. Here's an example of post-conditions for the process sale use case I have been using.

In practice, I often get questions about post-conditions from my clients. They think they may be doing something wrong, because they're just kind of regurgitating information that's already in the use case. For example, if we went back and read the nine or ten steps in the process sale use case we'd see specific steps for exactly what's in these post-conditions.

Post-conditions just summarize what will have been accomplished if everything works correctly. When my clients sometimes tell me that they aren't adding any value for them, I often tell them to drop the post-conditions since they don't add value. Some project stakeholders, however, like testers, may find the post-conditions to be valuable.

Another thing to remember about post-conditions is that a use case may not have any. Only when the state of a system is changed will there be post-conditions. As an example, if we had a use case that described a user doing a query of a database, there may not be any post-conditions for such a use case since the system state is not changed. If the query was logged, however, there would be a post-condition.