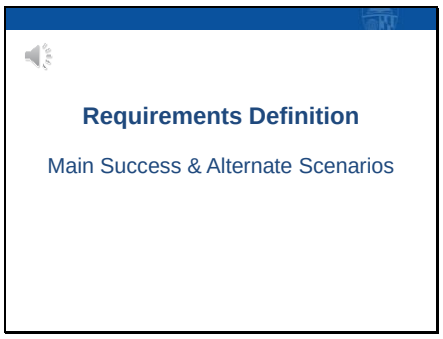
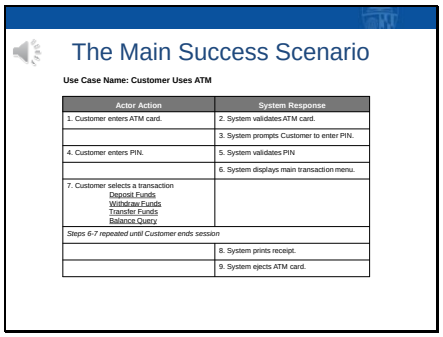
1



In this lecture we'll discuss how to write the main success scenario and the alternate scenarios for a use case.

2



Earlier, we discussed the basic information fields associated with a use case. The information fields dont' really contain any direct information about requirements. The requirements information is described in the use case scenarios.

The main success scenario is the first use case scenario that we write. It consists of a series of steps called action steps...and it also has a very important rule associated with it...everything works perfectly.

Here's an example of a main success scenario for a bank ATM system. If you read through the nine steps you can see that they are written under the assumption that everything works perfectly. Steps 2 and 5 are validation steps...and we assume that the validations are successful. We use a different type of scenario to deal with situations in which the validations are not successful.

Another basic guideline for writing a main success scenario is that it shouldn't have more than about 10-12 steps. The reason for this is that we want the reader of a main success scenario to be able to capture a mental picture of the basic story being told by this scenario...and if it has too many steps this will be hard to accomplish...there will be too much to remember.

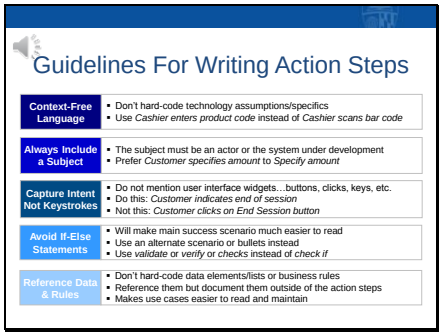
Take a look at each of the action steps. They take on certain characteristics. On the left hand column are the

actor interactions. The customer is interacting with the system. On the right hand side, the system is performing various functions...prompting, validating, and so forth. This is the only type of step we want to have on the right-hand side.

So...these illustrate two basic rules for action steps. An action step can be an actor interaction or a system function. A third type of action step is a branch to another use case...and that's illustrated in step 7. Depending upon the selection a customer makes the customer uses ATM story will branch to one of four other use cases...deposit funds, withdraw funds, and so forth. I'm underscoring the names of the use cases...it's just a notation I like to use because it's concise and the underscore connotes a hyperlink...which is exactly what branching to another use case is.

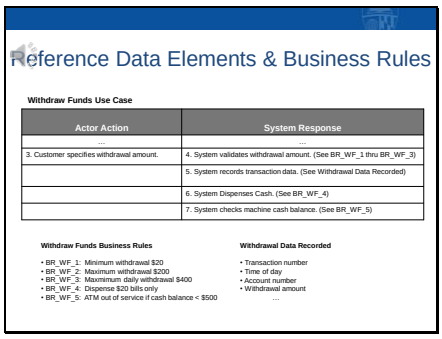
Note that below step 7 there is a statement written in italics. This is an annotation called a repetition clause. I wrote it in italics because I wanted it to stand out visually. Note also that this annotation is not numbered. Only the action steps should get numbered.

3



Here are a few guidelines for writing action steps. I'll elaborate on the last one…referencing data elements and business rules…and leave the others for your reference.

4

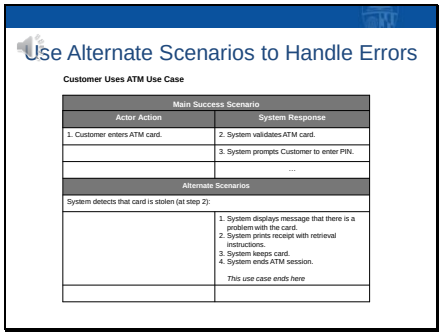


When business rules or data elements are associated with an action step, it's best to reference them in the action step rather than hard-coding them into the action step. This makes the use case more concise and easier to read, and improves maintainability in case the data elements or business rules need to be changed.

As an example, in step 4 of this use case a validation is performed. Validations are usually performed against business rules. I defined a set of five codes to correspond to the five business rules associated with this use case. I simply refer to the codes in the appropriate action steps...and I can put the business rules details either into a business rules field on the use case template or put them in the business rules section of a supplementary spec.

I did something similar in step 5. The system needs to store transaction data...so I referenced a list of the data elements that must be recorded. The details for data elements can be documented in a data elements field on the use case template or in a data elements section in the supplementary spec.

5



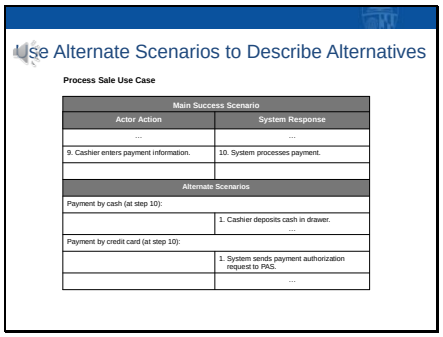
A use case has exactly one main success scenario, and it can also have any number of alternate scenarios. Alternate scenarios are written below the main success scenario. A use case always starts out on its main success scenario...and...depending on things that happen during the main success scenario...the story may branch off to one or more alternate scenarios. When the alternate scenario finishes, the story may branch back to the main success scenario, branch to another use case, or the use case may end at an alternate scenario.

One use of alternate scenarios is to handle errors. In this example, step 2 in the use case validates an ATM card. One type of error that might occur here is that the ATM card had been reported stolen...so we use an alternate scenario to describe what interactions are required to handle this type of error. Note also the annotation this *use case ends here* is used to tell the reader that the use

case ends if this scenario occurs.

Prior to the alternate scenario there is what we call a branching condition. It tells the reader why the story branched to the alternate scenario...in this example it branched because the system detected that the ATM card had been reported stolen...and it tells the reader where in the main success scenario the branching occurred...in this example at step 2.

6

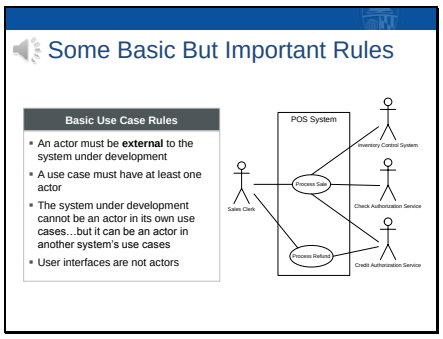


Alternate scenarios are also used to describe alternative ways the goal of a use case can be achieved. In a process sale use case, for example, we could have an alternate scenario to describe the different interactions that are required when a customer pays for merchandise using cash, credit card, and check.

Again...note that the branching conditions tell the reader two things: why the branching occurred and where it occurred. Unless both of these are specified a use case reader may not be able to follow it.

Branching conditions occur because the system detects things...like an incorrect PIN was entered...or because of some actor decision...like choosing to pay with a credit card, deciding to remove an item of merchandise from a sale...or because of an event...like a power failure.

7



In summary...it's important to remember the basic use case rules listed here. An actor must be external to the system we are defining requirements for, a use case must have at least one actor, the system itself can't be an actor in its own use cases, and user interfaces are not actors. There are additional rules, but for purposes of this course these will suffice.