**Use Case Testing:**

Use Case is a document which describes the end to end behavior of the system. It has “**User action**” which defines the action performed by the user and “**System action**” which describes the action done by the system.

The document describes the interaction of the user action and system action based on some transaction.

Both these actions are tightly coupled to each other. Because of some user action, there is / are some system action or vice versa.

Use case testing is a technique of deriving the test case through the use case document. We can also state that Use case testing is a technique of testing the whole system as a whole or doing the end to end testing based on transactions.

Use case testing tests with the vision of a user and not on the basis of input and output. It tests the real time transactions.

A use case document also has alternate flows written along with the expected transactions. While writing and executing the test cases, we should make sure that the alternate flows are also covered.

Use case testing is a type of complimentary testing and it does not provide the entire coverage of the application. You cannot judge the quality or decide the production deployment of an application, based on the use case test results.

Doing use case testing, we do and test the end to end flow of all the transactions (positive test case) and also cover the alternate flows (negative test case) based on the user action and system action. User action is the steps which the user does (Request) and system action is the output of user action (response).

**Use case testing focuses on the realistic data while doing the transaction.** We can use all the dynamic testing techniques to test the correct flows and the alternate flows. Use case testing should be used as a guideline for generating more detailed level testing.

Use case test cases are also supported by the preconditions which should be satisfied while executing the test case. Since it is based on the user / business view and realistic data, these preconditions forms an integral part of execution. Once the test case is executed, post condition can also be defined which defines the final state of the application.

# **What is Use case testing in software testing?**

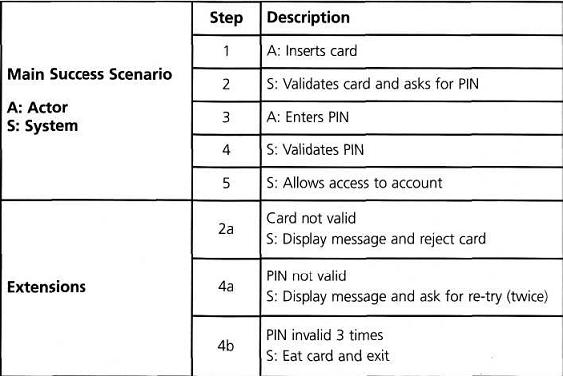
**Use case testing** is a technique that helps us identify test cases that exercise the whole system on a transaction by transaction basis from start to finish. They are described by Ivar Jacobson in his book Object-Oriented Software Engineering: AUse Case Driven Approach [Jacobson, 1992].

* A use case is a description of a particular use of the system by an actor (a user of the system). Each use case describes the interactions the actor has with the system in order to achieve a specific task (or, at least, produce something of value to the user).
* Actors are generally people but they may also be other systems.
* Use cases are a sequence of steps that describe the interactions between the actor and the system. Use cases are defined in terms of the actor, not the system, describing what the actor does and what the actor sees rather than what inputs the system expects and what the system’s outputs.
* They often use the language and terms of the business rather than technical terms, especially when the actor is a business user.
* They serve as the foundation for developing test cases mostly at the system and acceptance testing levels.
* Use cases can uncover integration defects, that is, defects caused by the incorrect interaction between different components. Used in this way, the actor may be something that the system interfaces to such as a communication link or sub-system.
* Use cases describe the process flows through a system based on its most likely use. This makes the test cases derived from use cases particularly good for finding defects in the real-world use of the system (i.e. the defects that the users are most likely to come across when first using the system).
* Each use case usually has a mainstream (or most likely) scenario and sometimes additional alternative branches (covering, for example, special cases or exceptional conditions).
* Each use case must specify any preconditions that need to be met for the use case to work.
* Use cases must also specify post conditions that are observable results and a description of the final state of the system after the use case has been executed successfully.

The ATM PIN example is shown below in Figure 4.3. We show successful and unsuccessful scenarios. In this diagram we can see the interactions between the A (actor – in this case it is a human being) and S (system). From step 1 to step 5 that is success scenario it shows that the card and pin both got validated and allows Actor to access the account. But in extensions there can be three other cases that is 2a, 4a, 4b which is shown in the diagram below.

For use case testing, we would have a test of the success scenario and one testing for each extension. In this example, we may give extension 4b a higher priority than 4a from a security point of view.

System requirements can also be specified as a set of use cases. This approach can make it easier to involve the users in the requirements gathering and definition process.

**[](http://istqbexamcertification.com/wp-content/uploads/2011/12/Use-case-testing-example2.jpg)**