1. What are the Requirement Gathering Techniques\methods used?

## Brainstorming – Used generally

1. Brainstorming is used in requirement gathering to get as many ideas as possible from group of people. Generally used to identify possible solutions to problems, and clarify details of opportunities.

## Reverse Engineering – Used for ROI WCF service securing

1. When a migration project does not have access to sufficient documentation of the existing system, reverse engineering will identify what the system does. It will not identify what the system should do, and will not identify when the system does the wrong thing.

## Survey/Questionnaire – Used in PRDTool at McAfee.

1. When collecting information from many people – too many to interview with budget and time constraints – a survey or questionnaire can be used. The survey can force users to select from choices, rate something (“Agree Strongly, agree…”), or have open ended questions allowing free-form responses. Survey design is hard – questions can bias the respondents.

## Document Analysis

1. Reviewing the documentation of an existing system can help when creating AS–IS process document, as well as driving gap analysis for scoping of migration projects. In an ideal world, we would even be reviewing the requirements that drove creation of the existing system – a starting point for documenting current requirements. Nuggets of information are often buried in existing documents that help us ask questions as part of validating requirement completeness

## Focus Group

1. A focus group is a gathering of people who are representative of the users or customers of a product to get feedback. The feedback can be gathered about needs/opportunities/ problems to identify requirements, or can be gathered to validate and refine already elicited requirements. This form of market research is distinct from brainstorming in that it is a managed process with specific participants.

## Interface analysis

1. Interfaces for a software product can be human or machine. Integration with external systems and devices is just another interface. User centric design approaches are very effective at making sure that we create usable software. Interface analysis – reviewing the touch points with other external systems is important to make sure we don’t overlook requirements that aren’t immediately visible to users.

## Interview

1. Interviews of stakeholders and users are critical to creating the great software. Without understanding the goals and expectations of the users and stakeholders, we are very unlikely to satisfy them. We also have to recognize the perspective of each interviewee, so that, we can properly weigh and address their inputs. Listening is the skill that helps a great analyst to get more value from an interview than an average analyst.

## Observation

1. By observing users, an analyst can identify a process flow, steps, pain points and opportunities for improvement. Observations can be passive or active (asking questions while observing). Passive observation is better for getting feedback on a prototype (to refine requirements), where active observation is more effective at getting an understanding of an existing business process. Either approach can be used.

## Prototyping

1. Prototyping is a relatively modern technique for gathering requirements. In this approach, you gather preliminary requirements that you use to build an initial version of the solution - a prototype. You show this to the client, who then gives you additional requirements. You change the application and cycle around with the client again. This repetitive process continues until the product meets the critical mass of business needs or for an agreed number of iterations.

## Requirement Workshops

1. Workshops can be very effective for gathering requirements. More structured than a brainstorming session, involved parties collaborate to document requirements. One way to capture the collaboration is with creation of domain-model artifacts (like static diagrams, activity diagrams). A workshop will be more effective with two analysts than with one.
2. What are the Use Case diagrams ?

An important part of the Unified Modeling Language (UML) is the facilities for drawing usecase diagrams. Use-cases are used during the analysis phase of a project to identify and partition system functionality. They separate the system into actors and use-cases. Actors represent roles that can are played by users of the system.

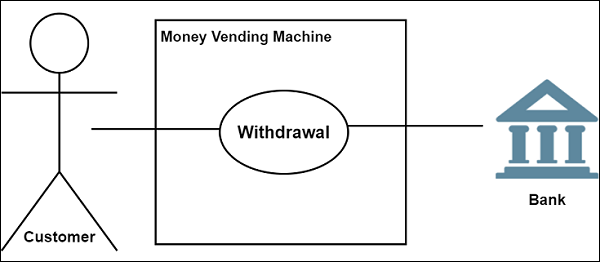
Those users can be humans, other computers, pieces of hardware, or even other software systems. The only criterion is that they must be external to the part of the system being partitioned into use-cases. They must supply stimuli to that part of the system, and the must receive outputs from it.

Use-cases represents the activities that actors perform with the help of your system in the pursuit of a goal. We need to define what those users (actors) need from the system. Use-case should reflect user needs and goals, and should be initiated by an actor. Business, actors, Customers participating in the business use-case should be connected to the use-case by association.

## Example ─ Withdrawal Use-Case

The goal of a customer in relation to our money vending machine (ATM) is to withdraw money. So, we are adding Withdrawal use-case. Withdrawing money from the vending machine might involve a bank for the transactions to be made. So, we are also adding another actor – Bank. Both actors participating in the use-case should be connected to the use-case by association.

Money vending machine provides Withdrawal use-case for the customer and Bank actors.



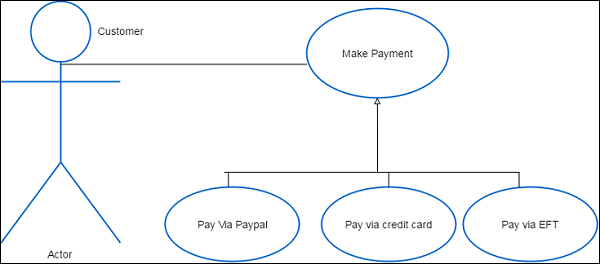
### **Relationships between Actors and Use-Cases**

Use-cases could be organized using following relationships −

* Generalization
* Association
* Extend
* Include

### **Generalization between Use-Cases**

There may be instances where actors are associated with similar use-cases. In such case a Child use-case inherits the properties and behaviour of the parent use. Hence we need to generalize the actor to show the inheritance of functions. They are represented by a solid line with a large hollow triangle arrowhead.



### **Association between Use-Cases**

Associations between actors and use-cases are indicated in use-case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use-case.

### **Extend**

There are some functions that are triggered optionally. In such cases the extend relationship is used and the extension rule is attached to it. Thing to remember is that the base use-case should be able to perform a function on its own even if the extending usecase is not called.

Extend relationship is shown as a dashed line with an open arrowhead directed from the extending use-case to the extended (base) use-case. The arrow is labeled with the keyword «extend».

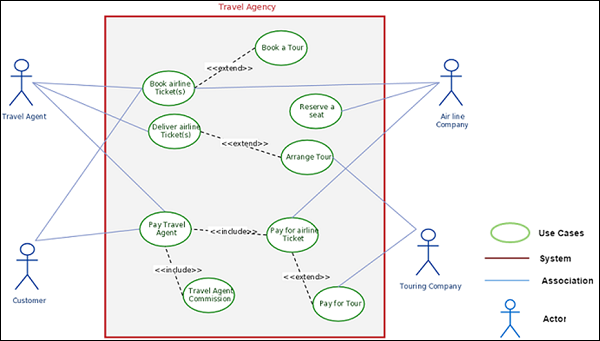
### **Include**

It is used to extract use-case fragments that are duplicated in multiple use-cases. It is also used to simplify large use-case by splitting it into several use-cases and to extract common parts of the behaviors of two or more use-cases.

Include relationship between use-cases which is shown by a dashed arrow with an open arrowhead from the base use-case to the included use-case. The arrow is labeled with the keyword «include».

Use-cases deal only in the functional requirements for a system. Other requirements such as business rules, quality of service requirements, and implementation constraints must be represented separately.

The diagram shown below is an example of a simple use-case diagram with all the elements marked.



### **Basic Principles for Successful Application of Use-cases**

* Keep it simple by telling stories
* Be productive without perfection
* Understand the big picture
* Identify reuse opportunity for use-cases
* Focus on value
* Build the system in slices
* Deliver the system in increments
* Adapt to meet the team’s needs

## Use-Case Template

Here, we have shown a sample template of a Use-Case which a Business Analyst can fill so that the information can be useful for the technical team to ascertain information about the project.

|  |  |  |  |
| --- | --- | --- | --- |
| Use-case ID: |  | | |
| Use-case Name: |  | | |
| Created By: |  | Last Updated By |  |
| Date Created: |  | Date Last Updated |  |
| Actor: |  | | |
| Description: |  | | |
| Preconditions: |  | | |
| Post conditions: |  | | |
| Priority: |  | | |
| Frequency of Use: |  | | |
| Normal Course of Events: |  | | |
| Alternative Courses: |  | | |
| Exceptions: |  | | |
| Includes: |  | | |
| Special Requirements: |  | | |
| Assumptions: |  | | |
| Notes and Issues: |  |  |  |