Topics: Software quality assurance, CMM, UML diagram, Use Case diagram

Software Quality Assurance

What is Quality?

Quality defines to any measurable characteristics such as correctness, maintainability, portability, testability, usability, reliability, efficiency, integrity, reusability, and interoperability.

There are two kinds of Quality:

Quality of Design: Quality of Design refers to the characteristics that designers specify for an item. The grade of materials, tolerances, and performance specifications that all contribute to the quality of design.

Quality of conformance: Quality of conformance is the degree to which the design specifications are followed during manufacturing. Greater the degree of conformance, the higher is the level of quality of conformance.

Software Quality: Software Quality is defined as the conformance to explicitly state functional and performance requirements, explicitly documented development standards, and inherent characteristics that are expected of all professionally developed software.

Quality Control: Quality Control involves a series of inspections, reviews, and tests used throughout the software process to ensure each work product meets the requirements place upon it. Quality control includes a feedback loop to the process that created the work product.

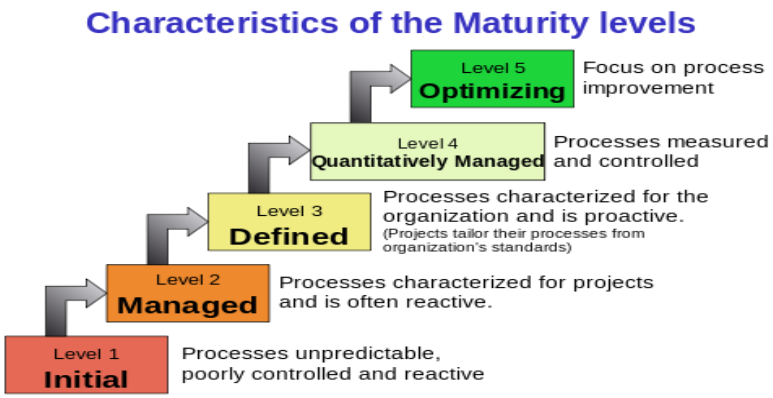
Quality Assurance: Quality Assurance is the preventive set of activities that provide greater confidence that the project will be completed successfully.

Quality Assurance	Quality Control
Quality Assurance (QA) is the set of actions including facilitation, training, measurement, and analysis needed to provide adequate confidence that processes are established and continuously improved to produce products or services that conform to specifications and are fit for use.	**Quality Control (QC)** is described as the processes and methods used to compare product quality to requirements and applicable standards, and the actions are taken when a nonconformance is detected.
QA is an activity that establishes and calculates the processes that produce the product. If there is no process, there is no role for QA.	**QC** is an activity that demonstrates whether or not the product produced met standards.
QA helps establish process	**QC** relates to a particular product or service
QA sets up a measurement program to evaluate processes	**QC** verified whether particular attributes exist, or do not exist, in a explicit product or service.
QA identifies weakness in processes and improves them	**QC** identifies defects for the primary goals of correcting errors.
Quality Assurance is a managerial tool.	Quality Control is a corrective tool.
Verification is an example of QA.	Validation is an example of QC.

Capability Maturity Model (CMM) & it's Levels:

Capability Maturity Model is used as a benchmark to measure the maturity of an organization's software process.

The CMM was developed at the Software engineering institute in the late 80's. It was developed as a result of a study financed by the U.S Air Force as a way to evaluate the work of subcontractors.



The entire CMM level is divided into five levels:

- **Level 1** (Initial): Where requirements for the system are usually uncertain, misunderstood and uncontrolled. The process is usually chaotic and ad-hoc.
- **Level 2** (Managed): Estimate project cost, schedule, and functionality. Software standards are defined
- **Level 3** (Defined): Makes sure that product meets the requirements and intended use
- **Level 4** (Quantitatively Managed): Manages the project's processes and sub-processes statistically
- **Level 5** (Optimizing): Identify and deploy new tools and process improvements to meet needs and business objectives

How long does it take to Implement CMM?

CMM is the most desirable process to maintain the quality of the product for any software development company, but its implementation takes little longer than what is expected.

- CMM implementation does not occur overnight
- It's just not merely a "paperwork."
- Typical times for implementation is
  - 3-6 months -> for preparation
  - 6-12 months -> for implementation
  - 3 months -> for assessment preparation
  - 12 months ->for each new level

Why Use CMM?

Today CMM act as a "seal of approval" in the software industry. It helps in various ways to improve the software quality.

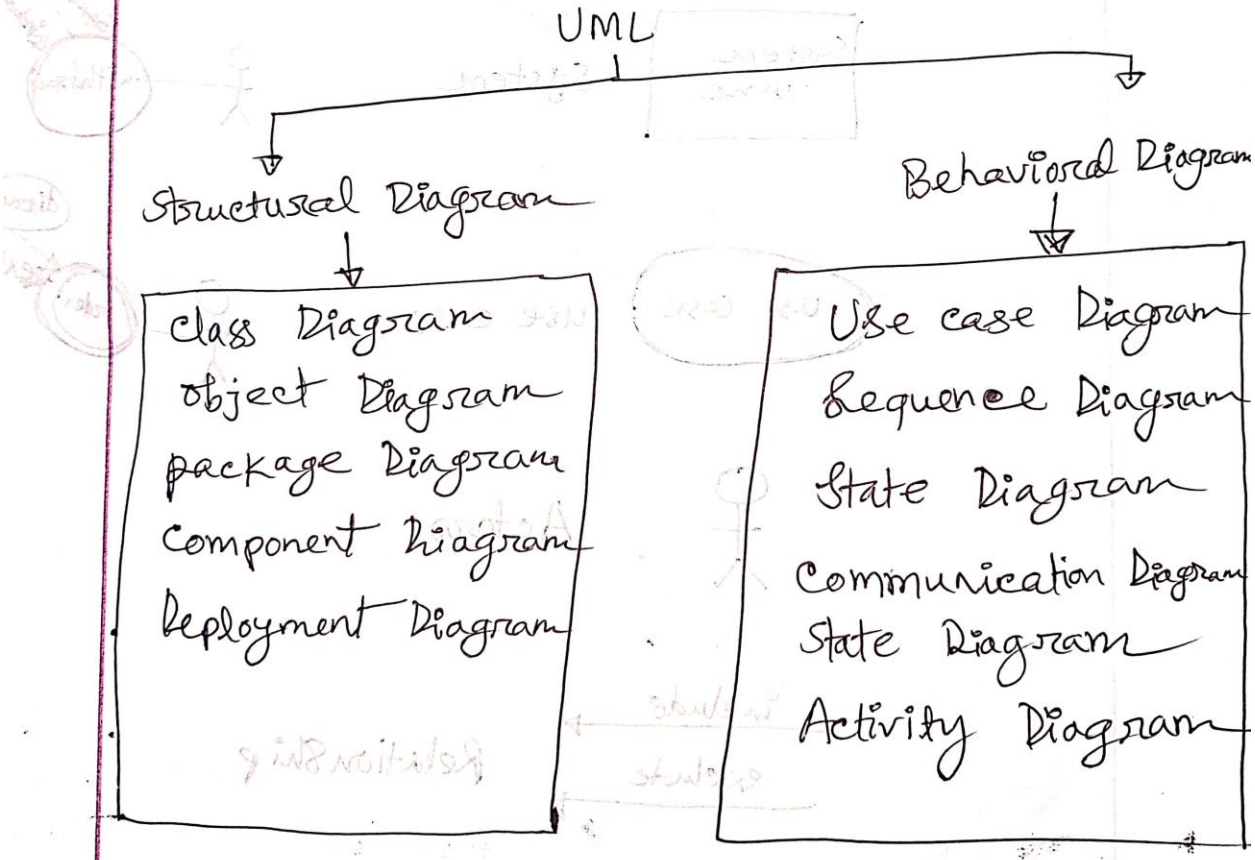
- It guides towards repeatable standard process and hence reduce the learning time on how to get things done
- Practicing CMM means practicing standard protocol for development, which means it not only helps the team to save time but also gives a clear view of what to do and what to expect
- The quality activities gel well with the project rather than thought of as a separate event
- It acts as a commuter between the project and the team
- CMM efforts are always towards the improvement of the process

UML diagram, Use Case diagram:

UML: Unified Modeling Language:

UML is a visual modeling language for analy-sis, design and implementation of software base system.

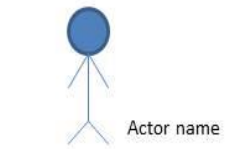
This language is used as a first step in deve-loping software and object oriented methodology



Use case:

Use case diagram is a behavioral diagram of UML.

Functional requirement of a system is visualized by use case.



Actor

Use case

Generalization symbol used between actors and between use cases

Association between actor and use case

Include relationship between use cases

Extend relationship between use cases

Symbols in a use case diagram

Use Case Diagram Relationships Explained with Examples

There are some relationship types in a use case diagram.

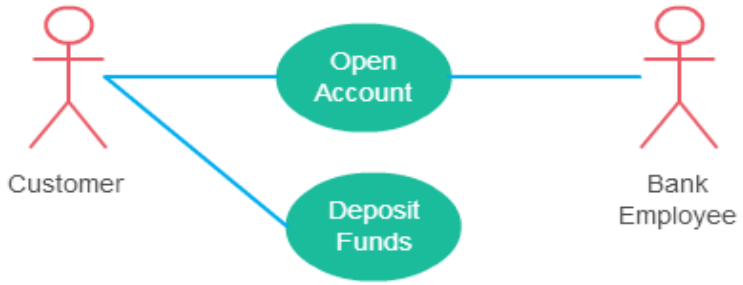
- Association between actor and use case
- Generalization of an actor
- Extend between two use cases
- Include between two use cases

Let's take a look at these relationships in detail.

Association between Actor and Use Case

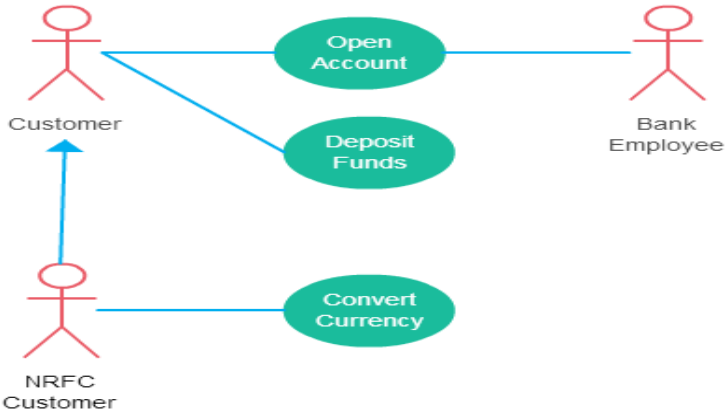
This one is straightforward and present in every use case diagram. Few things to note.

- An actor must be associated with at least one use case.
- An actor can be associated with multiple use cases.
- Multiple actors can be associated with a single use case.



Generalization of an Actor

Generalization of an actor means that one actor can inherit the role of the other actor. The descendant inherits all the use cases of the ancestor. The descendant has one or more use cases that are specific to that role. Let's expand the previous use case diagram to show the generalization of an actor.

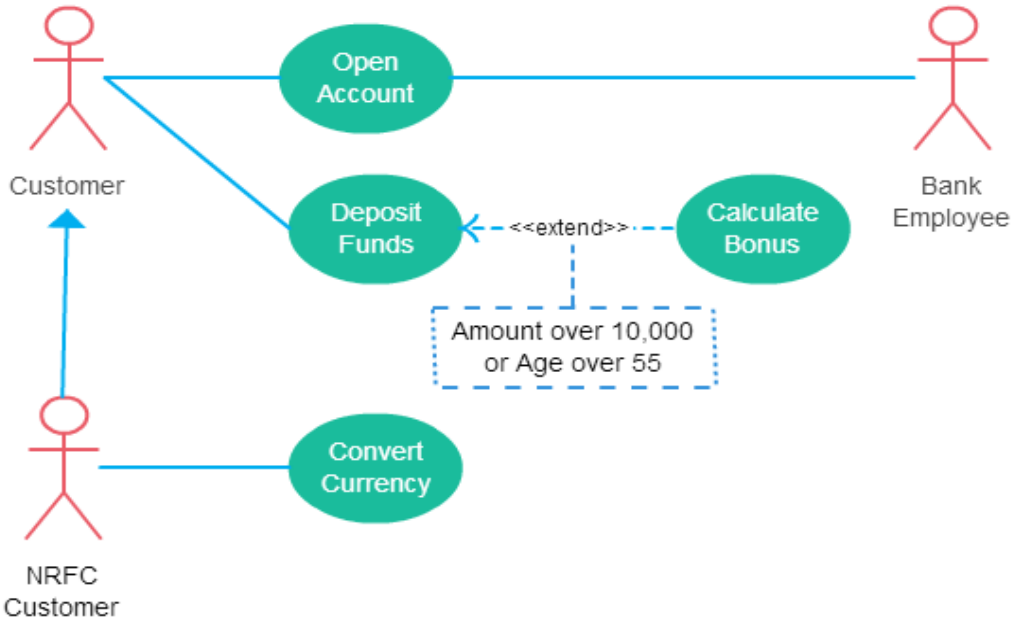


Extend Relationship between Two Use Cases

Many people confuse the extend relationship in use cases. As the name implies it extends the base use case and adds more functionality to the system. Here are a few things to consider when using the <<**extend**>> relationship.

- **The extending use case is dependent on the extended (base) use case**. In the below diagram the "Calculate Bonus" use case doesn't make much sense without the "Deposit Funds" use case.
- **The extending use case is usually optional** and can be triggered conditionally. In the diagram, you can see that the extending use case is triggered only for deposits over 10,000 or when the age is over 55.
- **The extended (base) use case must be meaningful on its own**. This means it should be independent and must not rely on the behavior of the extending use case.

Let's expand our current example to show the <<extend>> relationship.

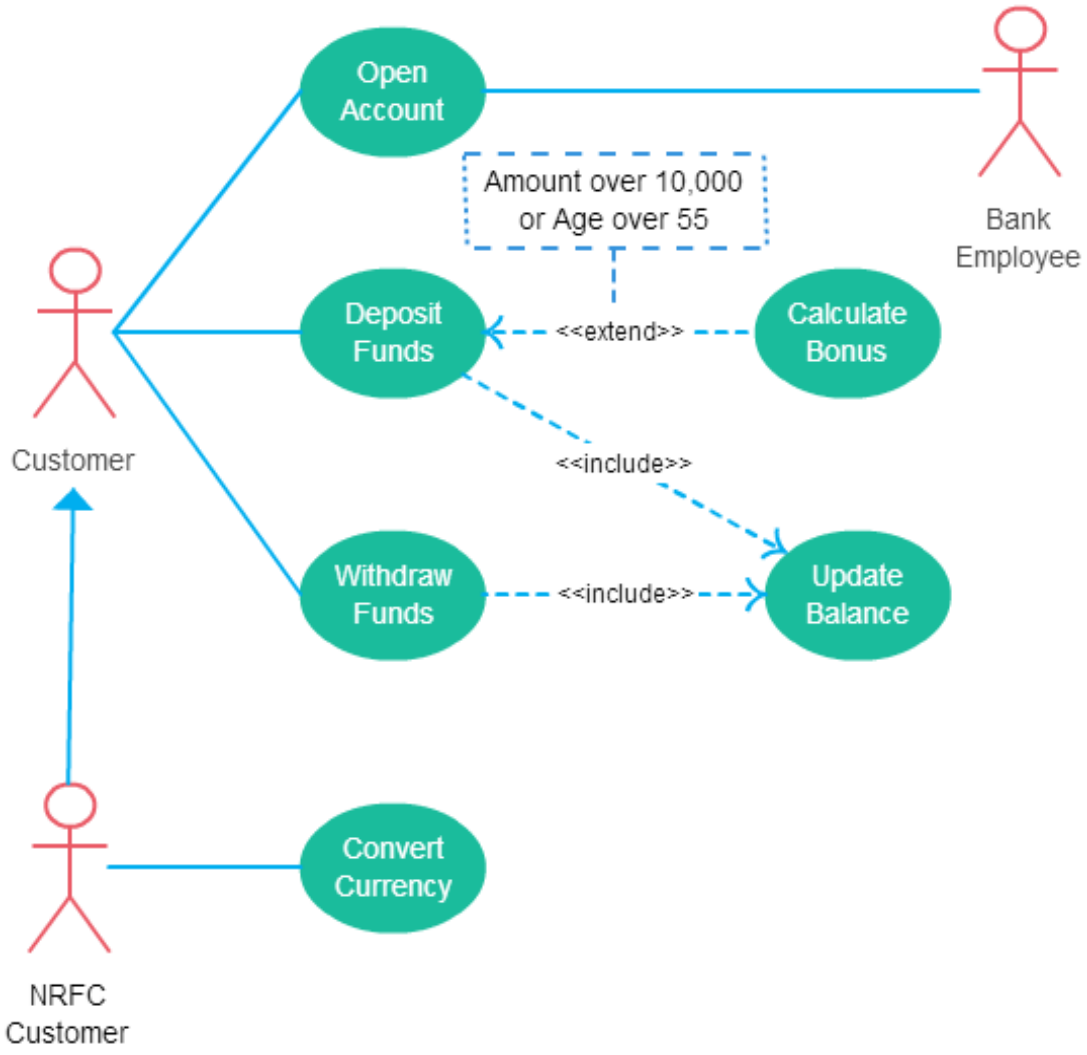


Include Relationship between Two Use Cases

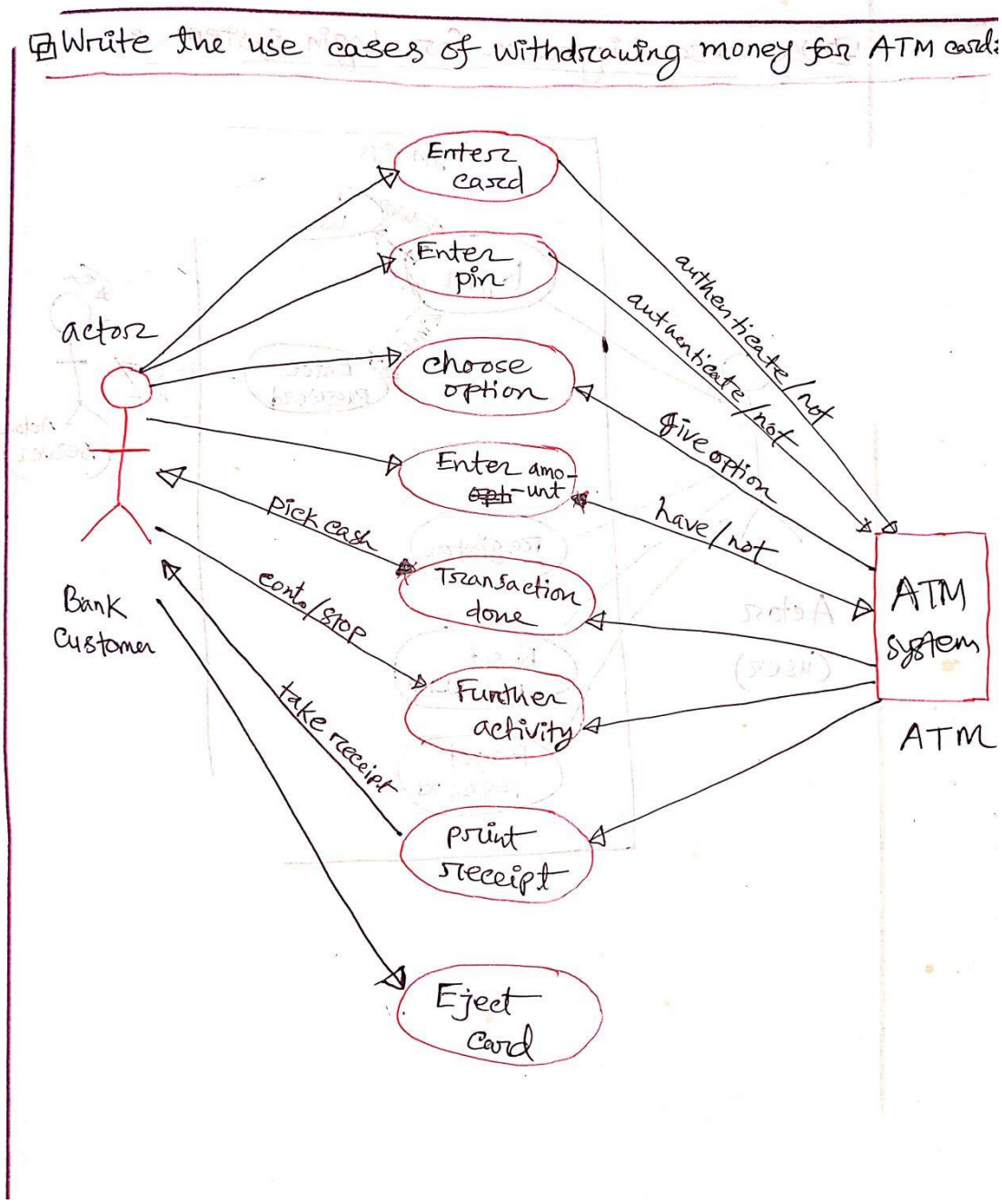
Include relationship show that the behavior of the included use case is part of the including (base) use case. The main reason for this is to reuse common actions across multiple use cases. In some situations, this is done to simplify complex behaviors. Few things to consider when using the <<include>> relationship.

- The base use case is incomplete without the included use case.
- The included use case is mandatory and not optional.

Lest expand our banking system use case diagram to show include relationships as well.


Open Account
Bank Employee
Amount over 10,000 or Age over 55
Deposit Funds
<<extend>>
Calculate Bonus
Customer
<<include>>
Withdraw Funds
<<include>>
Update Balance
Convert Currency
NRFC Customer

Example: Use Case



Use case Diagram for Login system :