 We have to exactly understand what customers actually wants. We will gather requirements and learn the entire process.

A Structured Business Analysis process will consist of the following: -

* **Understanding the Business** - What is the business is all about? They have to understand the problem.
* **Analysis Competition** - How the competition affects?
* **Analyzing the Market** - Every time the market is goanna be the same. Understand the market before finding any solutions.
* **Defining and scoping the project** - in Scope requirements - the list of requirements that your goanna include within a release.  Out Scope requirements - is what you don’t have to be a part of specific release it’s good if you have it shouldn’t be mandatory.
* **Gathering Requirements** - Business Analysts will be middle man between client and back-end side. SDLC critical part.
* **Analyzing and Documenting Requirements** - All of the documentations would be created by Business Analyst. (BRD - Basic Requirement Document AND FRD) Any of the industry they have a template for BRD and FRD.
* **Communication Requirements** - You are the person who is going back and forth.
* I**dentifying a Solution** -
* **Verifying that the solution meets the Requirements**

There are three key factors: -

 Documentation, Coordination and communication - For communication and coordination they hire a business analyst.

Definition - The **Business Analyst** is an agent of change. ... **Business analysis** is used to identify and articulate the need for change in how organizations work, and to facilitate that change. As **business analysts**, **we** identify and define the solutions that **will** maximize the value delivered by an organization to its stakeholders.

**The Role of Business Analyst**

* Business analysts are responsible for Identifying the business needs to their clients and stakeholders and to help determine solutions to business problems.
* A business analyst acts as a liaison between business people who have a business problem and technology people who know how to create automated solutions in an organization.
* They act as Business problem solvers.
* A business analysts’ main responsibility is to gather, detail and document requirements in a Format that is appropriate to the technical developers.

(High Level and Detail - High Level means Eagle Eye - What does Eagle do. Detail version is when they come to the ground.) - Like a broader perspective

* They provide the process questions and techniques to efficiently extract the information needed from the business users for successful development of projects.
* BAS is
  + Domain Independent (Healthcare, retail, mortgage, Airline, etc.)
  + Technology independent (Domain Specific E.g. Healthcare has facets, EDI transaction etc.)
* Gain Business Domain Knowledge - You have to gain knowledge if your project changes or what kind of industry you are providing solutions too.
* Understand Business Users - Satisfying the end users (Customers) in the primary Goal.
* Understand the Intricacies (Details) of an AS-Is System. (For example, if a company is using windows XP than that system is AS -is but to be system is let say they want to shift to windows 10.
* Propose Requirements to the TO-Be System by leveraging existing functionality.
* Negotiate between Business and Technology - some of the requirements may not be satisfied then you have to find solutions

(C level - Higher level like CEO)

* Address and resolve issues, document business decisions, manage change
* Envisage (Imagine) ‘END’ Vision - Be proactive about future state
* Adopt Customers to newer systems - Identify and promote customer benefits

**Need for Business Analyst**

Failed project factors

**incomplete requirements** - 13.1%

**Lack of user involvement** - 12.4%

lack of resources -10.6%

**unrealistic expectations** - 9.9%

Challenged project factors

**Lack of user input -** 12.8%

**incomplete requirements and specifications** - 12.3%            The Bolded one is where you work as BA.

c**hanging requirements and specification** - 11.8%

Business Analysis

Who is Business Analyst?

Why we have been Business Analyst (role)

A Business Analyst is the key figure in understanding business requirement and its implementation. This course is designed to help you understand Business Analysis right from Software Engineering Methods & Lifecycles to Requirements Preparation, analysis and presentation.

Business Analysts have emerged to have a key role in recent business scenarios. Some people think that the role of a Business Analyst is to make money for the organization, which may not be true in direct context. But indirectly, the action and decision taken by Business Analysts do leave an impact on the financial prospects of the organization.

Quality Analysis

Process to Examine the Quality Goals of a Project

Is any systematic process of determining whether a product or service meets specified requirements.

**A Structured quality analysis process will consist of the following**

* Ensuring the standards are followed for handling procedures.
* Preventing the server software quality assurance issues.

**Roles of QA**

* Two main roles: testing and documentation
* Software Developers Complete Software programming, Quality assurance analysis test it
* Uses a variety of tools, testing applications and reports to compare the software to user requirements
* Ensure that the software functions according to those requirements
* After testing, quality assurance analysts recommend changes and improvement for software
* Work with developers to make the appropriate changes and then test everything all over again.
* Document the Entire quality testing process
* Describe the program, evaluation, testing methods, and any improvements made to the software
* often create training materials and instruct users how to use the software properly.

**Need for Quality Analyst**

* Quality Analyst job during project lifespan is:
* Analysis of Quality design and improvising off tools.
* Checking the quality is matching up to client’s expectations and assuring the quality output of it.
* Once project is ready, they check the feasibility on various aspect.
* After Delivering getting a quality feedback and checking the quality of maintenance.

Domains with BA/QA Requirements

* Healthcare (>70%)
* Mortgage (>10%)
* Insurance (>10%)
* Retail (>10%)

What is Hardware/Software

* Hardware: is the physical aspect of computers, telecommunications, and other devices.
* Software: IS the term used to describe the various programs used to operate computers and other related devices.

Two main types of Software:

* Operating/system software
* Includes the programs that are dedicated to managing the computers itself, such as the operating system, file management utilities, and disk operating system. (OR DOS).
* Examples - Microsoft, Linux, Ubuntu

**Application Software:**

* Application software is specific to the task it is designed for and can be as simple as a calculator application or as complex as a word processing application.
* Examples- MS Office, Open office, Media, Players, MS Access, Educational Software, Media Development, Software, media development software, Antivirus Software, etc.

Front End:

* What can you see?

Back End:

* Refers to everything the user can’t see in the browser, like databases and servers
* Where the database is stored

Middle Tier:

* Many Layers of middle Tier. Various Interfaces that are connected to front and back end.

**Main types of Projects**

* **Greenfield Approach** - Greenfield is nothing, but a fresh project. Start from the base (From Scratch).
  + Who involve in Green Field Implementation?
    - PMO Office- Technical Team, SMES - Subject Matter Experts, Project Managers, IT Director and Program Managers.
    - Project Team- PM, BA, QA, Developers, Business Architectures.
      * *What you should know about greenfield implementation?*
        + Whole team is new to the Project
        + Team Members have very less idea of the project despite of they are having different level of prior experience.
        + It will take more time to get more knowledge of the project as PM has to give more and more information about the projects. E.g. Schedule more meetings.
        + BA has to act a crucial role when it comes to requirement gathering as business users might not have a clear understanding as well.
* **Legacy-up Gradation** - Upgrading an existing system. What is upgrading means? This simply means newer platform of the Software.
  + **Gap Analysis:** Gap Analysis is the quantitative and qualitative comparison of a company’s current performance against their potential. *As-Is to To-be process*.
  + Comparison of actual performance with potential or desired performance.
    - **AS IS- TO BE:** The old system - The new System
* **Integration/ Enhancement** - You can be hired in a project as a BA or QA to enhance the performance of the existing software
  + e.g. - AMAZON
  + Amazon website is already up and running, C- Level decides to add a feature to the existing system where customers can track their shipment with i the amazon website itself.
  + So, this is an enhancement Project. Add new feature to it.
  + Why? Competition add more benefits to the customers.
* **Data porting**- (Its more towards data analysis side)
  + Data porting is changing the server or the data where it is stored.
  + Extract the data, transform into what the new server can read. ETL (Extract, Transform and Load ) Change into the data dictionary of the other server.

Class Assignment

* Research about cloud computing models
* prepare a short presentation on types of cloud computing models

**Main Environment Types**

* **Development Environments**
  + Software Coding is done
  + It includes the technologies used for front end and back end.
  + It includes Development tools, Number of programmers, the organization policies.
* **Test Environments**
  + In which testing of software id one.
  + It comprises of circumstances or conditions influencing and effecting testing of a software
  + it also includes the organization policies, test tools, type of methodologies used no. of testers methods to improve testing.
* **Production Environments**
  + **T**he environment in which software is actually used by client.
  + It comprises of operating system of the client, browser used by client
  + Also referred to as real or live environment.

The Current trend among software development teams is to move away from QA- Based testing favor of developer- Based testing. But is that the right choice?

* Testing is the complete opposite. A tester’s task is to take simple features and think about the complex scenarios that will bring them down with bugs!
* We tend to be oblivious to flaws in our own creation and not been able to see the cases where one
* An important part of a tester's’ job is to [find the weak spots](http://www.amazon.com/exec/obidos/ASIN/B00HL2S4LW/makithecompsi-20) of the system and in it the breakable locations so they can be fixed. Most developers are concerned with solving problems, not creating them, which in turn makes it difficult for them to get into the tester’s mindset of destroying instead of building.

**Reason #1: Parental feelings towards their code**

**Reason #2: Developers simplify complex problems**

**Focusing on the “positive scenario”**

**Lack of End-to-End knowledge and real-user perspective**

Most organizations do not spend time and effort in order to make developers understand the users of their products. Costly developers are sent to do exactly what they were hired for, which is writing code and developing new products, and the task of understanding customers’ user stories is left for Product Managers.

Cloud Computing Basics

* The advancement of the technology encompassing networks
* storage
* processing power led to epitome of computing

What is cloud computing?

Cloud Computing is a Paradigm that allow on-Demand Network Access to shared Computing Resources.

A model For Managing, Storing and Processing data online via Internet.

Some cloud computing characteristics include:

Network Access

Uses Internet as a Medium

Resources are pulled together and allowed to use by multiple users.

SaaS - Software as a Service

On demand service

Pay per use of application software to users.

Independent platform.

Don’t need to Install the software n your PC.

Runs a Single Instance of the Software.

Available for multiple end users.

Cloud Computing Cheap.

Computing Resources Managed by Vendor

Accessible Via a Web Browser or Lightweight Client Application.

Pros

Universally Accessible from any platform

No need to Commute, you can work from anyplace

Excellent for Collaborating working

Vendor Provides Modest Software Tools

Allows for multi tenancy

Cons

Portability and Browser Issues

Internet Performance May Dictate Overall Performance

Compliance Restrictions

Used by End Customers

PAAS - Platform as a service

This Service is Made up of a programming Language Execution Environment, An Operating System, A web Server & A Database.

Encapsulate the environment where users can Build, Compile and Run Their programs without worrying of the underlying infrastructure.

In this Model, You Manage Data & The Application Resources; All other resources are Managed by the Vendor.

Developers Users of PAAS

Amazon Web Services

Heroku

[force.com](http://force.com/)

Pros

Cost Effective Rapid Development (it’s Scalable

Faster Market for Developers

Easy Deployment of web applications

Private or public deployment is possible

Cons

Developers are limited to the providers Languages and tools

Migration issues - such as the risk of vendor lock - in

IAAS

Infrastructure as a service

This service offers the computing architecture and infrastructure, all computing resources but in a Virtual Environment so that Multiple Users can access them. Resources Include; Data Storage, Virtualization, Servers and Networking.

Most Vendors are responsible for managing the above four resources.

User will be responsible for handling other resources such as applications, data runtime and middleware

Sysadmins

E.g.

Amazon EC2

Go Grid

Rackspace

The cloud provides the infrastructure

Enhanced scalability - dynamic workloads are supported

IAAS is flexible

Cons

Security Issues

Network and Service delays

Amazon [AWS]

Cloud Computing

Elastic bean stock

iCloud - back up and store everything online

Microsoft Azure by Microsoft

Office 365

Google Cloud Universal platform for googles vast ecosystem

IBM smart cloud provides full range of IAAS SAAS.

**SDLC - Software Development life cycle**

Software Development Life Cycle -

1. Planning
2. Requirements Analysis
3. Design
4. Implementation and Coding
5. Testing <- This is you
6. Deployment
7. Maintenance

Defining the Application

1. User Registration
2. Login
3. Logout
4. Dashboard Landing Page

Software Development Life Cycle- A Framework defining tasks performed at each step in the software development process.

SDLC includes 5Ds and 1M:

1. Discover
2. Design
3. Develop
4. Debug
5. Deploy
6. Maintenance

**Discover/Analyze Phase**

This phase is about researching and defining the scope of the project it involves activities such as:

* Establishing Business goals
* Understanding what a successful outcome would be
* Carrying out User research
* Mapping the customer journey
* Reviewing the competition

Discover

* Company
  + Concept/ Idea by C- Level
    - Budget by C- Level
      * Steering Committee “The Steering committee provides support, guidance and oversight of progress (VP & C level)
      * PMO “To oversee the entire project” (IT Director, PM, SME, Technical Team)
      * Project team (PM, BA, QA, Developers, Technical Team)
      * Documents
        + Kick-off presentation- is the first meeting with the project team and the client of the project
        + Project charter - is the statement of the scope, objective and participate of the project
        + project plan-formal approved document used to guide both project execution and project control.

**Components of kick off presentation**

* Idea/ Concept
* Budget
* List of people involved
* Roles & Responsibilities matrix
* Project Milestone
* Project Timeline

What is Project Milestone?

A milestone is a marker in a project that signifies a change or stage in development. Milestones are powerful components in project management because they show key events and map forward movement in your [project plan](https://teamgantt.com/guide-to-project-management/how-to-plan-a-project/).

Milestones act as signposts through the course of your project, helping ensure you stay on track. Without project milestone tracking, you’re just monitoring tasks and not necessarily following the right path in your project. Milestones can do more than just show progress—they can help you communicate what’s happening with your project.

The milestone plan charts the logical ties or dependencies between milestones. The milestone plan shown in Exhibit 1 should be understood as follows: to reach a state set out in milestone plan M5, the state described by milestone M4 must be in place. It is logically impossible to reach milestone M5 before M4. A milestone plan is therefore a logical plan. It shows the logical interconnections between milestones.

**Components of Project Charter responsibility of project Manager**

* Scope
* Objectives
* Roles & Responsibilities Matrix
* Version History
* Budget Estimation
* System
* Timeline and Milestone
* Deliverables

**Components of Project Plan**

**Created by project Manager with the approval of PMO Office**

* **Components**
* Date
* Task
* Resources
* Progress
* Escalation Point/ Process

**Key terms in SLDC Process**

* Project Charter:
  + The most important result of the project initiation phase is the project charter, which formally authorizes the work of the project to being (Or continue) and give the project manager the authority to do his job.
* Project Scope:
  + The project scope is the definition of what the project is supposed to accomplish and the budget of both time and money that has been created to achieve these objectives.
* Change control board (CCB):
  + A committee that makes decisions regarding whether or not proposed changes to a software project should be implemented.

Class Assignment

* Create a Project Charter

RACI Chart/TOOL

* A matrix of all the activities or decision-making authorities undertaken in an organization set against all the people or roles.
* R Responsible
  + who is/will be doing the task? - Who is assigned to work on the task?
* A: Accountable
  + Who will be accountable? Whose head will go if things go wrong.
* C: Consulted
  + Anyone who can tell them more about the task. Any stakeholders already identified
* I: Informed
  + Who need to be informed?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Process/ Phase | R | A | C | I |
| KOP | PM | PM | Steering Committee | C-Level |
| Project Charter | PM | PM | Steering committee | Steering Committee |
| Project Plan | PM | PM | PMO | Steering Committee |
| BRD  FRD  Test Plans | BA  BA  QA | PM  BA  QA | PM/Business team  PM  PM | PMO  PM  PM |

**Abbreviation**

KOP - Kick of Presentation

BA- Business Analyst

PC \_ project Charter

PP- Project Plan

BRD- Business Requirement Document

FRD - Functional Requirement Document

SRD- Specific Requirement Document

TRD- Technical Requirement Document

RTM.UML - Requirement Traceability matrix Unified Modeling Language

QA - Quality Analysis

* Test plans
* Test Scenario
* Test Cases/ Test Script
* Test Data
* UAT - User acceptance testing

Design Phase

* In the design phase the architecture is established.
* This phase starts with the requirement document delivered by the requirement phase and maps the requirements into an architecture
* The architecture defines the components, their interfaces and behaviors.
* This phase represents the “How” phase/
* Details on computer programming languages and environments, machines, packages, applications architecture, Distributed architecture layering, memory size, platform, algorithms, data Structures, Global Type Definitions, interfaces and many other engineering details gets established.

Design/Components are as follows:

* BRD (Business Requirements Document)
* FRD (Functional Requirements Document)
* SRD (System requirements document)
* TRD (Technical Requirements Document)
* Wireframe and mockups
* RTM (Requirements Traceability Matrix)
* TRM (Requirement traceability Matrix): It is a document that maps and traces user requirement with test cases. The main purpose of requirement traceability matrix is to see that all test cases are covered so that no functionality should miss while doing software testing.
* It’s a document that is delivered once development/Testing is completed, so it should be a living document, it essentially maps your requirements to your test cases so that you have traceability of your coverage as you test whatever in development.
* A **test Case** is a specification of the inputs, execution conditions, testing procedures, and expected results that define a single test to be executed to achieve a particular software testing objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

MS Visio, Jira a bug tracker

Business Requirement Document (BRD)

* High -Level Business of the organization developing the product or solution with the help of it.
* The BRD Template contains-
  + Table of content
  + Project Scope
  + Objective/ Overview (Specifically for the document what re the things that are included in the document)
  + In/ Out Scope
  + Roles and Responsibility Matrix
  + Version History
  + Sign Off/ Approval
  + Requirements - Functional/ Non- Functional

Functional Requirement Document (FRD)

* This document talks about the functionality of the software in detail it is usually created by business analyst under the supervision of technical expert, for instance system architecture.
* The FRD template contains-
  + Introduction- It should contain purpose, scope, background, references, assumption and constraints, document overview.
  + Methodology
  + Functional Requirements
  + Modeling Illustrations- Context, user requirements, data flow diagrams, logical data model/ Data dictionary, functional requirements.
  + Other Requirements - Interface requirements, Hardware/Software requirements
  + Glossary

System Requirement Document (SRD)

A system requirement specification (SRS) (Also now as a software requirements specification) is a document or set of documentation that describes the features and behavior of a system or software application.

* introduction
  + Purpose
  + Scope
  + Definitions, Acronyms and abbreviations
  + References
  + Overview
* Overall Description
* Specific Requirements
  + Functionality
  + Usability
  + Reliability and availability                    “Non Functional
  + Performance
  + Security
  + Supportability
  + Design Constraints
  + on- Like User documentation and help system requirements
  + Purchased Components
  + Interfaces
* User Interfaces
  + Hardware interfaces
  + Software interfaces
  + Communication Interfaces
* Licensing Requirements
  + Legal, Copyright and other notices
  + Applicable standards
* Supporting information

**Technical Requirement Document (TRD)**

A technical requirement pertains to the technical aspects that your system must full-fill, such as performance-related issues, reliability issues, and availability issues.

* Revision Log
* Executive Summary
* Assumptions, Risks, and Dependencies
* Requirements
  + Functional Requirements
  + Non-Functional Requirements
* Reference
* Glossary

**Functional and non-functional Requirement**

Functional Requirement- Functional Requirements describe **what the system must do**.

* A Functional Requirement will describe a particular behavior of function of the system when certain conditions are met, for example send email when a new customer sign ups or open a new account
* For example, if you consider a shopping site, adding items to cart, browsing different items, applying offers and deals and successfully placing orders comes under functional requirements.
* Nonfunctional requirements any requirement which specifies how the system performs a certain function.
* Non- Functional requirements generally specify the systems quality attributes or characteristics. For example, “modified Data in a a database should be updated for all users accessing it within 2 seconds.
* Whereas performance of the system in peak hours, time taken for the system to retrieve data from DB, Security of the user data, ability of the system to handle if large number of user’s login comes under nonfunctional requirements.

Difference between business requirement document and functional requirement document

Difference between system requirement document and technical requirement document

Wire-Frame- MS Visio and for Diagram -

Development Phase

The development phase features a key step in the project: system construction. The previous phases lay the foundation for system development: The Following phases ensure that the product functions as required. To complete the development phase successfully, two elements are required: -

1. A complete set of design specifications
2. Proper processes, standards and tools

Activities in development phase: -

* Coding
* Performing Unit Testing

1. Deploy

* Software Deployment is the activity that make a software system available for use. The phase includes deployment preparation and procedures, product deployment, transferring ownership of the product, and closing the deployment phase.
  + Implementation plan
  + Production support

1. **Maintenance**

* Validating The issue
* Temporary fix
* Permanent Solution

SLDC Methodologies

* Waterfall
* Agile

BOLT

SQL

w3 school

SQL BOLT

database administrator- DBA

Quality Center- QC

Structured Quality Language -SQL

**SDLC Methodologies**

**Waterfall**

**Agile**

**Waterfall Methodology**

* First SDLC model
* The waterfall model is a linear sequential (Non-iterative) design process, used in software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases
* Simple to understand and easy to implement

Different phases within waterfall

Requirement Gathering: IN this phase business analyst will collect requirement with an integration of client collected requirements will be documented

Requirement Analysis: In this phase system analyst will study the client requirements and prepare the system requirements specification

Design: In this phase design architecture is the responsible to decide architecture of an application in order to full fill the client requirements

Coding: In this phase developers will write the program using programming languages or scripting languages in order to develop the application

Testing:  Initially developers will perform unit testing and integration testing using of white box testing. After that separate team will be perform system testing using black box testing

Release and maintenance: after the testing client satisfied on work product then we deliver application to the customer to use at live environment while using the application client identify can same defect in existing system then he will send to the CR to CCB.  (CR= change control and CCB= change control board)

Waterfall methodology chart

Requirement- Requirement Doc

                        Prepare use case

Design- Software architecture

            Map the stakeholders

Implementation: Construct the software

                        Data storage and retrieval

Verification:    Install

                        Test and Debug

Maintenance:  Check errors

                        Optimize capabilities

Waterfall methodology phases (SDLC)

Discover (product owner, director, BA, PM, PO)

DESIGN (BA, PM, PO, DESIGN ARCHITECT, LEAD QA)

DEVELOP (Developers, BA, PM, PO, Lead QA)

Debug (QA, BA, PM, PO)

DEPLOY (LEAD DEVELOPERS, BA, PO, PM)

MAINTAIN

PROCESSWS WITHIN WATERFALL (SDLC)

A PROJECT CHARTER IS CREATED UPON THE COMPLELTION OF THE KICK OFF PRESENTATION PHASE, WHICH DESCRIBES THE SCOPE, OBJECTIVE FOR THE PROPOSED PROJECT

ONCE THE PROJECT CHARTER IS DRAFTED, A FORMAL APPROVED DOCUMENT AS PROJCET PLAN IS DOCUMENTED

THE BUISNESS ANALYST WIL RECORD ALL REQUIREMENT DOCUEMNTS

ONCE THE DESIGN IS SIGNED OFF, THE PROJCECT MOVES ON TO THE DEVELOP STAGE

ONCE THE DEVELOPMENT OF THE PROJCET IS COMPLETED, IT WILL BE SIGNED OFF BY A LEAD DEVELOPER FOR THE PROJECT TO MOVE ON TO THE DEBUG STAGE

A GO/NO GO MEETING IS HELD DURING THE DEBUG STAGE WHERE THE PM, PO, LEAD DEVELOPER, LEAD QA, AND BA DISCUSS HOW TO PROCEED ON THE PROJECT AND APPROVE IT FOR PRODUCTION (SELL IT TO THE CUSTOMER)

(Whether you have to move with the production of software or not)

**Advantages**

* As scope is well defined in, it is easier to focus on production.
* Simple and easy to understand
* Product definition is stable in waterfall
* Requirements are well known, clear and fixed
* Phases don’t overlap, phases are processed and completed one at a time.

**Disadvantages**

* Cost of fixing error is high as once the cycle has started there is no way going back. Iteration has to start from beginning for debugging
* Difficult to go back and change something
* Not suitable for product with high risk of changes
* No working software until late stages of SDLC
* risk and uncertainty
* Not suitable for complex project
* Focus more on documentation rather than working software

Agile is a set of values and principles

Daily stand up

Collection of believe developing software

Agile Manifesto

Doing it and helping others do it

**What is Agile?**

Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. Agile methods generally promote a disciplined project management process that encourages frequent inspection and adaptation. A leadership philosophy that encourages teamwork, self-organization and accountability. A set of engineering best practices intended to allow for rapid delivery of high-quality software, and a business approach that aligns development with customer needs and company goals. Agile development refers to any development process that is aligned with the concepts of the agile manifesto.

Agile approaches help teams respond to unpredictability through incremental, iterative work and practical feedback.

The Agile Manifesto (SDLC)

**The Agile Manifesto defines four guidelines:**

* **Individuals and interactions** over the processes and tools
* **Working software** over comprehensive documentation
* **Customer collaboration** over contract negotiation
* **Responding to change** over following a plan.

-Each of the guidelines should be read keeping in mind the following:

“That is, while there is value in the items on the right, we value the items on the left more” (WISDOM AND COMMON SENSE OVER KNOWLEDGE AND EXPERTISE)

**Agile Umbrella**

|  |  |
| --- | --- |
| Light - weight  Approaches | Fuller Approach |
| Scrum | DSDM (Dynamic Systems Development Method) |
| Lean | Agile Unified Process |
| Kanban | FDD (Feature Driven Development) |
| Crystal |  |
| Extreme- Programming (XP) |  |

**Agile Umbrella**

* SCRUM - Scrum is an agile way to manage a project, usually software development. Agile Software development with scrum is often perceived as a methodology; but rather than viewing scrum as methodology, think of it as a framework for managing a process.
* Kanban- It is an agile process framework. Kanban is a method for managing the creation of products with an emphasis on continual delivery while not overburdening the development team. Like scrum, Kanban is a process designed to help teams work together more effectively.
* XP- Extreme programming is a software development methodology which is intended to XP describes four basic activities that are performed with the software development process: coding, testing, listening and designing.
* RUP (Rational Unified Process)

**SCRUM Approach** (AGILE SDLC)

* A light-weight agile process tool.
* Split your organization into small, cross-functional, self-organizing teams.
* Split your work into a list of small, concrete deliverables. Sort the list by priority and estimate the relative effort of each items.
* Split time into short fixed- Length interactions/sprints (Usually 2-4 weeks)
* Optimize the release plan and update priorities in collaboration with the customer, based on insights gained by inspecting the release after each iteration.
* Optimize the process by having a retrospective after each iteration.

**Things we do in SCRUM**

* The project/ Product is described as a list of features: The backlog
* The features are described in terms of user stories
* The scrum team estimates the work associated with each story.
* Features in the backlog are ranked in order of importance.
* Result: A ranked and weighted list of product features.
* Daily Scrum meeting to discuss what did you do yesterday? What will you do today? Any obstacles?

As a user I want this so that I can do this (Format for user stories)

**SCRUM Values**

* Commitment - Team members individually commit to achieving their team goals, each and every sprint.
* Courage - Team members know they have their courage to work through conflict and challenges together so that they can do the right thing.
* Focus: Team members focus exclusively on their team goals and the sprint backlog; there should be no work done other than through their backlog.
* Openness: Team Members and their stakeholders agree to be transparent about their work any challenges they face.
* Respect- Team members respect each other to be technically capable and to work with good intent.
* Product Owner - The product owner is responsible for creating and maintaining product backlog and acceptance criteria documents
* Scrum Master- The scrum master is responsible for coordinating and managing day to day work and to isolate the scrum team from outside interference.
* Scrum Team- The Scrum team is responsible to execute the work planned in each release or sprint.

**Sprint**

* Short time where certain functionalities of the project are developed within that time period. (Last up to 2 -4 weeks in general, and can last up to 5 -6 weeks if complex)
* Stand-Up meetings are held after the project starting date, and before the start of each work day after to discuss; what you did the previous day, what you will do today, and what your work plan is for tomorrow.
* Prioritizing product backlog.

**Backlog**

* List of things that need to be done maintained by the product owner (PO)

**Types of backlog**

* Product backlog
  + simply a list of all things that need to be done within the project.
  + This is a wish list describing end product/ Application in terms of epics and user stories.
* Sprint Backlog
  + Is a list of tasks identifying by the scrum team to be completed during the scrum sprint?
  + Sprint backlog contains number of user stories, scrum team is going to work on in current, / upcoming sprint/ iteration.
  + During the sprint planning meeting, the team selects some number of product backlog items, usually in the form of user stories and identifies the tasks necessary to complete each user story.

**Agile Buzzwords**

**Sprint BURNDOWN Chart:**

The sprint burndown chart is a graph that depicts the amount of the work remaining at the start of each sprint.

**EPIC:**

    Epic is collection of two or more user stories. AS project progress, each epic will be further refined in user stories. Epic is a large body of work which contains a lot of stories and will be completed in multi sprints. Epic is parent level issue type of story and task.

**User Story:**

* A user story is a format used in Agile software development to capture a description of a software feature from an end-user perspective.
* The user story describes the type of user, what they want and why.
* A user story helps to create a simplified description of a requirement (Actor, Scenario)
* They are short, Simple description of features told from the perspective of the user/customer
* User real world scenarios to document the user needs.
* User stories are easy to read.

Example:

    “As a <type of User> I < want/can/ am able to/ need to/ ETC> So that < Some reason>”

Task (In Jira)

 Task is a piece of technical works should be done within a single sprint by engineers.

Sub- Task (In JIRA)

    SUB-Task is child level issue type of story and task, which is the work break down of story and task.

**Acceptance Criteria:**

* Acceptance criteria are the conditions that a software product must satisfy to be accepted by a user, customer, or in the case of system level functionality, the consuming system.
* Acceptance criteria document is one of the scrum artifacts, which explains how desired functionality must perform.
* Validates the user stories and epics.

**Acceptance criteria format:**

* The “Given/When/Then” Format is helpful way to specify criteria:
  + Given I am on home page
  + When I Pass in my credential
  + And I click sign in Button
  + Then I should be on my account Page
  + Precondition when I do some action then I expect some result
  + When writing acceptance criteria in this format it provides a consistent structure.

**SCRUM Poker:**

* Sprint poker is a discussion/ Meeting where a deck of agile cards are used to discuss the progress of a sprint or a future sprint.
* To start a poker planning session, the product owner or customer reads an agile user story or describes a feature to the estimators.
* Each Estimator is holding a deck of planning poker cards with values like 0,1,2,3,5,6,13,20,40 and 100.

**Daily scrum meeting/ Daily stand-up meeting:**

This is daily meeting where the scrum team discusses what they did yesterday, what they are going to do today and what are obstacles they are facing the duration of this meeting is 15 mins and it occurs daily.

**Sprint Planning meetings:**

This meeting is conducted in order to finalize the number of user stories which are going to cover in next sprint along with estimation. If length of your sprint is one week, sprint planning meeting duration would be 2 hours and it occurs at the start of the sprint.

**Sprint Review Meeting:**

 This meeting is conducted to review completed sprint/iteration with customer. If the length of your sprint is one week, the duration of sprint review meeting would be 1 hour, and it occurs at the end of the sprint.

**Sprint retrospective meeting:**

This meeting is conducted to analyze the process followed by scrum team and to identify areas of improvements. If length of the sprint is one week, the duration of your retrospective meeting would be 45 mins and it occurs at the end of the sprint.

BA’s Role in SCRUM

* Work closely with the product owner and other stakeholders to elicit requirements
* Work closely with the product owner and the scrum master to support product backlog
* work with the product owner to prioritize requirements/ Backlog
* Maintains user stories and epics based on the elicited requirements
* Participate in meetings
* coordinate with the development team to help them understand the business logic behind requirements

Class Assignment

Differentiate between SCRUM and KANBAN

Prototype

* The activity of creating prototypes of software applications, for example, incomplete versions of the software program being developed.
* It is used to visualize some component of the software to limit the gap of misunderstanding the customer requirements by the development team.

Type of prototypes

Working Prototype:

* A dummy version of the software with working functionalities such as links, drop down menu’s etc.

Non- Working Prototype:

* A dummy version of the software with no working functionalities, just the layout.

**What is a requirement?**

* A condition or capability needed by a stakeholder to solve a problem or achieve an objective.
* It must meet or possess a solution component to satisfy standard, specification, or other formally imposed documents.
* It is about what not how?
* Requirements are the descriptions of the services provided by a system and its operational constraints/ Restrictions.
* It may range from a high-level abstract statement to a detailed mathematical specification
* It may be as complex as a 500 Page Description.

**Requirement Engineering**

* It is the process of discovering, analyzing, documenting and validating the requirements of the system.
* Each software development process goes through the phase of requirements engineering.
* Requirements Engineering activities include
  + Elicitation - Gather of the requirements
  + Analysis
  + Specification
  + Verification

**Types of Requirements**

* User Requirements
* System Requirements

User and System Requirements

**User Requirements**

* Written for customers
* Often in a natural language, no technical details

**System Requirement**

* Written for developers
* Detailed functional and non-functional requirements
* clearly and more rigorously way

**User Requirements**

* Usually the first attempt for the description of the requirements
* Services and constraints of the system
* Written in natural Language of Diagrams
* Readable by everybody
* Serve business objectives

**System Requirements**

* Lack of Clarity: Ambiguous Language
* Requirements Confusion functional, non-functional requirements, design information is not distinguished
* Requirements amalgamation: Several Requirements are defined asa single one (Combining one requirement with other requirements)
* Incompleteness requirements may be missing
* Inconsistency requirements may contradict themselves.
* Services and constraints of the system in detail
* user for the design and development
* Precise and cover all cases
* Structured presentation
* Elaborate the user requirements to get a precise, detailed and complete version of them
* Used by designers and developers

Example of User + System Requirement

* User Requirement- The PAC system should allow images to be taken of the patient. (Picture archiving and communication system)
* System Requirement- The PAC System should allow the specialist to take images of the patient of particular part. It will not allow if the patient is not in the member LIST-IF the patients has no appointment-if the patient has not taken necessary precautions prior the procedure.

Classification of Requirements

**Business Requirements:** - Define the goals and objectives of the business at the enterprise level.

**Stakeholder Requirements**: - Describe the goals and objectives of a particular group within an organization.

**Solution Requirements**: - Describe the various characteristics of a solution that must be met.

**Transition Requirements:** - Describe any capabilities of the solution that aren’t permanent but instead exist only to facilitate the transition from the current state to the future state.

**Functional + Non- Functional Requirements**

Functional Requirements - Describe the behavior and information that the solution will manage (I/P, O/P such as passwords and log in Details)

Non-Functional Requirements - Describes the qualities of the process or system (Product requirements, organizational requirements, external requirements)

**Non- Functional Requirements Examples**

* Reliability requirements such as load times, lag times
* Interoperability requirements such as whether software are workable on different platforms or portals to conduct business transactions
* Standard Requirements such as whether the software meets the standards followed and set forth by the client.
* Privacy requirements such as who has access to what within the software.
* Usability requirements such as if layout of the software is sure friendly.

Measure of Non- Functional Requirements

|  |  |
| --- | --- |
| Property | Measure |
| Speed | Processed transactions, second user, event response time, screen refresh time |
| Size | Giga byte, mega bytes |
| Ease of use | Training time, number of incidents |
| Reliability | Mean time to failure, probability of unavailability, rate of failure occurrences, availability |
| Robustness | Time to restart after failure, percentage of events causing failure, probability of data corruption in case of failure |
| Productivity | Percentage of target dependent statements, number of target systems. |

Elicitation of non-functional requirements

* Stakeholder goals, values and concern
* Legacy system and/or existing platform constraints.
* Competitive analysis of system qualities - AS-IS - To Be
* Industry and market trends.
* Standard non-functional requirements templates and categories
* Pre- Established Trigger questions (Trigger reason to have the project)

Guidelines to Minimize Issues with Requirements

* Separate Requirements:
  + Separate the requirements, separate functional and non-functional requirements, requirements must be clearly identified (e.g.: By a number or bullet points)
* Include a rationale for each Requirement:
  + Helps clarify reasoning behind the requirements and may be useful for evaluating potential changes in the requirements.
* Invent or use Standard Form/ Template.
* Distinguish Requirements Priorities (Example - Moscow = Must, Should, Could , Wont)
  + As an account holder I must be able to login in my account
  + As a n account holder my ID should have alphanumeric
  + As an account holder my account’s home page should be in purple color
  + As an account holder my account balance should be visible before I login to my account
* Avoid Technical Jargon
* Testable (Write test Cases)
* Deliverables (Know what time the end results should be)

**Feasibility Study**

* It is done at first to decide whether or not the project is worthwhile
* Look at different perspectives (Market Analysis, Financial ,Schedule, technical, Resource, Legal ,Etc.)
* Should make you aware of the risks.
* Doing the study
  + Consult information sources: Managers, software engineers, end users, etc.
  + based on information collection (Interviews, Surveys, questionnaires etc.)

Feasibility Study

* What if the system wasn’t implemented?
* What are current process problems?
* Do technical resources exist?
* What is the risk associated with the technology?
* Is new technology Needed? if so, what skills?
* How will the proposed Project contribute to the overall objectives of the organization?
* have the benefits identified with the system being Identified?
* What will be the integration problems?
* What facilities must be supported by the system?
* What is the risk associated with cost and schedule?
* What are the potential disadvantages/ advantages?
* Are there legal issues?
* Are there issues linked with the fact that this is an offshore project?

**Requirements**

* A thing that is needed or wanted
* A thing that is compulsory: a necessary condition.

**Business Requirements**

High-level business goals of the organization developing the product or solution with the help of it.

**Requirement Analysis**

Requirements analysis, also called requirements engineering, is the process of determining user expectations for a new modified product.

**Scope/ Requirement Creep**

Scope Creep (Also called requirement creep; or kitchen sink syndrome) in project management refers to changes, continuous or uncontrolled growth in a project scope, at any point after the project begins. This can occur when the scope of a project is not properly defined, documented or controlled.

**Requirement Elicitation and Analysis**

**Stakeholders:** Person or group of persons who will be affected by the system (Directly or Indirectly)

All Stakeholders should be involved in requirements elicitation and analysis.

* Stakeholders express requirements in their own terms
* Different stakeholders may have conflicting requirements
* organizational and political factors may influence the system requirements.
* The requirements change during the analysis process. New Stakeholders may emerge and the business environment change.

**BA’s main role after requirements elicitation**

* Requirements documentation
* Requirements validation.
* Requirements Management.

**Babok Requirement elicitation techniques**

* Brainstorming
* Interview
* Observation
* Prototyping
* Focus Group
* Process modeling
* Requirements workshop/ JAD
* Document Analysis
* Interface Analysis
* Surveys/ Questionnaires

**Brainstorming: -**

* Brainstorming is used in requirements elicitation to get as many ideas as possible from a group of people.
* Generally used to identify possible solutions to problems and clarify details of opportunities.
* Brainstorming casts a wide net, identifying many different possibilities
* Prioritization of those possibilities is important to finding the needles in the haystack.

**Interview: -**

* This is commonly used and valued technique.
* Interviews of stakeholders and users are critical to creating the great software.
* It's beneficial to start with unstructured interview in order to gain the current work environment or the basic need of the project.
* Later on, structured interview is conducted with prepared set of questions.

**Observation: -**

* The study of users in their manual habitats is what observation is about.
* By observing users, an analyst can identify a process flow, awkward steps, pain points and opportunities for improvement.
* Observation can be:
  + Active: - Asking questions while observing
  + Passive: - Doesn’t ask questions just makes notes.

**Document Analysis: -**

* Reviewing the documentation of an existing system when creating as-is process documents, 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: -**

* A focus group is 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.
* There is danger in “following the crowd” and some people believe focus groups are at best ineffective.
* One risk is that we end up with the lowest common denominator features.

**Interface Analysis: -**

* 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 marking sure that we create usable software.

**Prototyping:**

* Prototypes can be very effective at gathering feedback.
* Low fidelity prototypes can be used as an active listening tool.

**Requirements workshop/ JAD sessions**

* More commonly known as join application design /development session workshops can be very effective for gather requirements.
* More structured than a brainstorming session, involved parties collaborate to document requirements and develop solutions.
* It consists of series of workshops where a team of key players (Business and IT)
* One way to capture the collaboration is with creation domain-model artifacts (Like stick- Diagrams, activity Diagrams)
* A workshop will be more effective with two analysts that with one, where a facilitator and a scribe work together.

**Survey: -**

* A survey can be used to collect information when you are faced with problem f having too many people to interview and face budget and time constraints.
* It is a good technique to gather info from remote locations.
* Ask for follow-up questions
* Ask for documents needed
* The survey can force users to select from choices, rates something (Agree strongly, agree, disagree), or have open ended questions allowing free-from responses.
* A well-designed survey would provide qualitative guidance for characterizing the market.

**Babok Requirement Elicitation Techniques**

* **Joint Application Development (JAD):**
  + JAD is used as a technique for developing business system requirements
  + The purpose of JAD is to bring together it and the business community in a structured workshop setting to extract consensus - based system requirements. This accomplished by using a trained JAD Facilitator and customized, planned Agendas to assist the participants in arriving at complete, high quality requirements. Experience has shown that the JAD process substantially reduces development time, costs and errors.
  + JAD is typically used in the early stages of a project life cycle

**STEP 1** Planning Ahead

* Planning begins long before JAD sessions start. IT team represented by the PM meets with the executives to establish-
  + The scope and limitations of the project
  + Clear, Understandable, measurable and attainable goals
  + The expected milestones, deliverables and timelines
  + Planning continues during actual JAD sessions
  + Prior to each meeting agenda is developed, agreed upon and distributed to the participants
  + Minutes are distributed of previous session to the participants for review.

SMART - Specific, Miserable, Attainable, Realistic, Time Bound

**STEP 2:** Assembling the right team

* Executive Sponsor - Is a higher-Ranking person in the business org who establishes the project vision and scope and selects business
* Facilitator - Serves as a guide, directing the flow of the discussions, eliciting participation from the team members, resolve conflicts and maintains the focus of the sessions, should have excellent communicating skills and heavily experience in facilitating the success of JAD session.

**USER/ Participants -**

* **Strategic users**- They are supervisory or management positions who understand the business policies.
* **Real End users-** They are the people who will ultimately use the new system in their day to day work.

Combination of both types provides valuable insight for the development of the new system.

* **IT Representatives: -**
  + Cross section of the IT team is present at the sessions to learn about the business, offer advice, ensure that the clients will be able to understand technological constraints and capabilities and help develop solution.
* **Scribe -** Main function is to record and publish the proceedings in a timely manner.

**STEP 3-** Ensure everyone is committed to the project

**STEP 4 -**

* At the conclusion of the session IT team will produce the design document within the timeframe agreed upon. The team will present the solution to the executive sponsor and demonstrate the prototype.
* During This stage the team obtains the executive approval to proceed and the entire team gathers together to evaluate the JAD process.

**Requirements analysis scenarios**

* IT is often easier for people to relate on real-life examples rather than abstract statements.
* Scenarios are description or sequences of events of how the system used in practice.
* Scenarios are composed of:
  + A description of the initial state of the system.
  + A description of the normal flow of events in the scenario.
  + A Description of what can go wrong and how it is handled.
  + Information on other activities that might be going on concurrency.
  + A description of the final state of the system.

**USE CASE**

* A written description of the User’s interaction with the software product to accomplish a goal.
* It is an example of behavior of the system.
* Written from an actor’s point of view, not the systems.

Use case is a documentation technique to identify, clarify, and organize system requirements to achieve a goal using a set of steps and one or many actors.

Use case define what will system do.

Use cases define the interaction between the system and its users.

Use case defines a set of scenarios that describe externally visible behavior that delivers value to one or more users of the system.

**UML Use case Diagram**

**Use Cases:**

* A use case are a list of action or event steps, typically defining the interaction between a role and the system.
* Use cases identify the users of the system (Actors)
* Use cases identify the tasks (Related to the system).
* Use cases relate the users and the tasks.
* Use case are typically illustrated with UML as stick figures or similar Diagrams.
* A set of use cases should describe all possible interactions with the system
* Use cases are more effective in capturing functional requirements.

**The textual Description of a use case includes**

* A use case ID
* Use case name
* actors
* Description
* pre-condition
* Post- condition
* Normal course
* Alternative course
* Extends

**Use cases Terminology**

* Actors- The people or external computer systems that will communicate with the system.
* Goals- the things that the actors want to achieve
* Use case title: Variation in the verbiage of the main goal
* Use case body: A bunch of things that describe the requirements of the system and its actors.

**Use cases have relationships**

* **Inclusions:**
  + A use case contains the behavior from another use case (unconditional)
  + Can be seen as a factorization
  + Introduced by the <<include>> keyword

(Checkout) ————> (Scan Item)

                   —————> (Calculate total and tax)

* **Extensions:**
  + A possible branch in a use case scenario, often triggered by an error or failure in the process.
  + Useful for finding edge cases that need to be handled and tested.
  + Introduced by the <<Extends>> Keyword.

Use cases are broken into four elements

* System - website, software components, business process, application etc.
* Actors
* Use cases
* Relationship

**What is UML?**

What is UML stands for Unified Modeling Language. It’s a rich language to model software solutions, application structures, system behavior and business processes.

The UML combines the best features of

* Data modeling concepts (Entity relationship Diagrams)
* Business modeling (work flow)
* Object Modeling
* component modeling

It can be used all processes, throughout the development life cycle and across different implementation technologies.

**UML Goals**

* The primary goals in the design of the UML were: provide users with a ready-to-use expressive visual modeling Language so they can develop and exchange meaningful models.
* Define an easy-to-learn but semantically rich visual modeling language
* Provide a formal basis for working with models
* Provide a visual way of building and exchanging models
* Provide ways to extend the core concepts
* Be independent of programming language and development processes
* Support Higher-level repeatable and reusable development concepts
* Frameworks patterns and components
* Integrate best practices

**Types of Diagrams**

* **Structural Diagrams**
  + Class/ Object Diagrams- show the static class structure, function and contents
* **Behavioral Diagrams**
  + Activity Diagrams - Focuses on flow driven by processing
  + Use Case Diagrams- shows the relationship among actors and use cases , defined the business context.
  + State Diagrams- Describe the dynamic interactions between entities that they manipulate thus activity diagrams are realized
* **Interaction Diagrams**
  + Sequence Diagrams- Display the time sequence of component interactions
  + Collaboration Diagrams- Display components and their links among one another
* **Physical Diagrams**
  + Component Diagrams- Shows the High- Level Structure of the code itself
  + Package Diagrams- Shows the hardware organization and decomposition of subsystems
  + Deployment Diagrams - shows the configuration of elements at runtime.

**Use Case Diagram**

* Depicts the functionalities of the system as users perceive it
* Provides a high-level overview of a system
* A use case is a unit of business functionality
* Illustrates who will use the system, what they will be able to achieve with the system
* Excellent way to communicate to management, development team, testing team, end users and other non-development people about what the system will do.
* ’Sets context’ for conversations about requirements and system development.
* Starting point when designing a new system using the UML
* Other UML diagrams evolve from the User Case model.

A use case diagram is a collection of actors, use cases, and their communications

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

**A Use Case Diagram Notation**

|  |  |  |
| --- | --- | --- |
| **Actor** | An Actor is a user of the system depicted using a stick figure.  The role of the user is written beneath the icon.  Actors are not limited to humans. If a system communicates with another application, and expects input or delivers output, then that application can also be considered an actor. |  |
| **Use Case** | A use Case is functionality provided by the system, typically described as verb + object (e.g. Register Car, Delete User)  Use Cases are depicted with an ellipse. The name of the use case is written within the ellipse. |  |
| **System Boundary** | Marks the bounds of the system.  Use cases are drawn inside the system boundary, actors are placed outside. |  |
| **Package** | Logical folders to group related use cases together. |  |
| **Directed Association** | Associations are used to indicate actors - Use case and use case - use case interactions.  Directed associations are depicted by an arrow with. the direction indicating the direction of flow of data. |  |
| **Association** | Associations are used to indicate Actors- Use case and Use case - Use Case interactions.  (Undirected) associations are depicted by a line indicating flow of data in both directions. |  |
| **Generalization** | Inheritance relationship (generalization in UML) is indicated by a block in-shaded arrow from the child class (actor or use case) pointing to the parent class (actor or use case). |  |
| **Dependency** | Indicates the dependency between two use cases. Arrow point towards dependent use case |  |
| **<<Include>>**  **stereotype** | A base use case executes <<included>. use case as part of its flow to complete its business functionality |  |
| **<<Extend>>**  **stereotype** | A base use case may be extended by an <<extend>> use case based on a certain condition. |  |

**Use Case Diagram**

* Pictorially depicts the functionalities of the system, the people and external systems (actors) interacting with the system.
* Drown at the system level representing multiple use cases, use case- use relationship, actors - use case relationship
* Visualizes functional requirements

**Use Case**

* Textually describes the interactions between the system and the actor.
* Is a sequence of related transaction performed by an actor and the system in a dialogue?
* Documents functional requirements

**Actors**

An actor is someone or something outside the system that interacts in the system under development (SuD)

Could be a human, external system, service, subsystem or time

Examine actors need to determine use cases.

Primary Actors

* Initiate use cases

**Secondary Actors**

* Provide inputs
* Receive messages
* Act as Services

**<<include>> Construct**

As the use cases are documented, other use case relationships may be discovered,

* A uses relationship shows behavior that is common to one or more use cases.
* An extends relationship shows optional behavior.

**<<Extend>> Construct**

Signifies that one-use cases are <<extend>> the functionality of other

E.g. B extends A

* A is the base use case, B <<extend>> the functionality of A
* B is run only when a certain condition(s) is/ are satisfied when A runs
* A can run (produce success outcome) without running B
* A may call B multiple times during a flow

Usually used to keep special/ specific business cases separate from a specific flow.

**Use Case Format**

Title - must start with a verb - Make Payment

Description and purpose

Diagrams and documents

Pre-conditions

Post-conditions

Included/ extending use cases

Glossary

Primary Actors

Secondary Actors

Main Flow

Alternate flow(s)

Exception flow(s)

**Activity Diagram**

Shows the structure of a process or other computations as the flow of control and data from step to step within the computation.

* Activity Diagrams Models Shows:
  + Workflow and behavior
  + Similar to the old data flow diagram
* Create an activity diagram to:
  + Demonstrate the logic of an algorithm
  + Describe the steps performed in a UML use case
  + Illustrate a business process or work-flow between users and the system.
  + Simplify and improve any process by clarifying complicated use cases.
  + Model software architecture elements such as method, function and operation.

**Basic Components of an Activity Diagram**

* **Action -** A step in the activity wherein the users or software perform a give task
* **Decision Mode**- A conditional branch in the flow that is represented by a diamond. It includes a single input and two or more outputs.
* **Control flow** - Another name for the connectors that show the flow between steps in the diagram.
* **Start Node -** Symbolize the beginning of the activity. The start node is represented by a black circle.
* **End node** - Represented the final step in the activity. The end node is represented by an outlined black circle.

**State Diagrams**

**State Diagrams-** Notes changes of state in the system

    Add State, Activity and Transition

A state transition diagram shows

* The life history of a given class
* The events that cause a transition from one state to another
* The actions that result from a state change

State transition Diagrams are created for objects with significant dynamic behavior.

* Shows a state machine, consisting of states, transitions, events, and Activities.

A screen shot of a computer

Description automatically generated

A close up of a map

Description automatically generated

**Interaction Diagram**

* The use case Diagram presents an outside view of the system
* Interaction Diagrams describe how use cases are realized as interactions among societies of objects.
* Two types of interaction Diagrams
  + Sequence Diagrams
  + Collaboration Diagrams
* Sequence Diagrams and collaboration Diagrams are used to demonstrate the interactions of components in a use case.

**Sequence Diagram**

Sequence diagrams describe interactions among classes (functions) in terms of an exchange of messages over time. They’re also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modeling a new system.

A screen shot of a computer

Description automatically generated

**Introduction to Testing**

* Software testing is the process of analyzing a software item to detect the differences between existing and required conditions (That is , Bugs) and to evaluate the features of the software item.
* Software Testing is an activity to check whether the actual results match the expected results and to ensure that the software system is defect free.
* Software testing also helps to identify errors, Gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools.
* Software testing is really required to point out the defects and errors that were made during the development phases.
* It’s essential since it makes sure of the customer’s reliability and their satisfaction in the application.
* It is very important to ensure the quality of the product. Quality product delivered to the customers helps in gaining their confidence.
* Testing is necessary in order to provide the facilities to the customers like the delivery of high-quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results.
* Testing is required for an effective performance of software application or product.
* It is important to ensure that the application should not result into any failures because it can be very expensive in the future or in the later stages of the development.

**Testing Terminology**

* **Quality** - can be defined as a “Degree of excellence”. high quality software usually conforms to the user requirements.
* **Quality Assurance -** is set of processes used to measure and assure the quality of product.
* **A test Plan** - is a document dealing a systematic approach to testing. A system such as a machine or software. The plan typically contains a detailed understanding of what the eventual workflow will be.
* **Test Case** - is a description of what to be tested, what data to be given and what actions to be done to check the actual result against the expected result.
* **Test Data -** Data that exists (for example, in a database) before test is executed and that affects or is affected by the component or system under test.
* **Traceability Matrix** - Is a document, usually in the form of a table that correlates any two base lined documents that require to determine the completeness of the relationship. It is often used with high-level requirements of the software product to the matching parts of test cases.
* **Defect -** is referred as if software misses some features or function from what is there in requirements.
* **Bug -**is referred as a fault in a program which causes the program to perform in an unintended or unanticipated manner.
* **Showstopper -** bug is hardware or software bug that causes an implementation to stop and become essentially useless. This critical bug must be fixed for the development process to proceed further.

**Software Testing Life Cycle (STLC)**

* Software testing life cycle refers to a testing process which has specific steps to be executed in a definite sequence to ensure that the quality goals have been met. In STLC process, each activity is carried out in a planned and systematic way. Each phase has different goals and deliverables. Different organizations have different phases in STLC; However, the basis remains the same.
* It consists of series of activities carried out methodologically to help certify your software product.

1. Requirements Analysis
2. Test Planning
3. Test Case Development
4. Environment Setup
5. Test Execution
6. Test Cycle closure

**Role of QA during the project life cycle**

|  |  |
| --- | --- |
| **Project Phase** | **Role of the Quality Analyst** |
| Analysis Phase | Develops the Test Plan and understand the requirements |
| Design Phase | Starts writing the test cases. |
| Development | Completes the test cases and verifies the ambiguities |
| Test | Executes the test cases |

**Requirement Analysis**

Requirement Analysis is the very first step in software testing life cycle (STLC). In this step quality assurance (QA) team understands the testable requirements. If any requirements have conflicts or are missing out or not understandable than QA team follows up with the various stakeholders and cross function team for better understanding of requirements. QA is involved in the STLC from the very first step which helps to prevent the defects into software under testing the requirements can be either function or non- Functional like performance, security and testing.

|  |  |  |
| --- | --- | --- |
| Entry Criteria | Activities | Deliverable |
| Following documents should be available:   1. Requirements Specification 2. Application Architecture   Along with both Acceptance criteria should be well defined. | Prepare the list of questions or queries and get resolved from Business Analyst, System Architecture, Client, Technical Manager/Lead etc.  Make out the list for what all Types of Tests performed like Functional, Security, and Performance etc.  Define the testing focus priorities,  List down the test environment details where testing activities will be carried out.  Checkout the Automation feasibility if required & prepare the Automation feasibility report. | List of questions with all answers to be resolved from business i.e. testable requirements Automation feasibility report (if applicable) |

**Test Planning**

Test Planning is most important phase of software testing life cycle where all testing strategy is defined. This phase is also called as test strategy phase. In this phase typically QA lead based on the organization is involved to determine the effort and cost estimates for entire project. This phase will be kicked off once the requirement gathering phase is completed. Based on the requirement analysis, Test plan is created. The result of test planning phase will be Test Plan or Test Strategy & testing Effort Estimation Documents.

|  |  |  |
| --- | --- | --- |
| **Entry Criteria** | **Activities** | Deliverable |
| Requirements Documents (updated version of unclear and missing requirements. | Define Objective & Scope of the project.  List down the testing types involved in the STLC.  test effort estimation and resource planning.  Selection of testing tool if required  Define the testing process overview.  Define the test environment required for entire project.  Prepare the test schedules  Define the control procedures.  Determining roles and responsibilities.  List down the testing deliverable.  Define the entry criteria, suspension criteria, resumption criteria and exit criteria.  Define the risk involved if any. | test plan or Test strategy document. Testing Effort estimation document. |

**Test Case Development**

The test Case Development activity is started once the test planning activity is finished. This is the phase of STLC where testing team write down the detailed test cases. Along with Test Cases Testing team also prepare the test Data if any required for testing. Once the test cases are ready then these test cases are reviewed by peer members or QA lead. Also, the requirement traceability matrix (RTM) is prepared. The requirement Traceability Matrix is an industry accepted format for tracking requirements where each test case is mapped with the requirement. Using this RTM, we can Track backward & forward traceability.

|  |  |  |
| --- | --- | --- |
| **Entry Criteria** | **Activities** | Deliverable |
| Requirements Documents (updated version of unclear or missing requirement)  Test Plan signed off and approved | Preparation of test cases.  Preparation of test automation scripts (if required).  Prerequisite test data preparation for executing test cases. | Test cases  Test data  Test Automation Scripts (if required). |

**Test Environment Setup:**

Setting up the test environment is Vital part of the STLC. Basically, test environment decides on which conditions software is tested. This is independent activity and can be started parallel with test case development. In process of setting up testing environment test team is not involved in it. Based on company to company may be developer or environment admin creates the testing environment. Meanwhile testing team should prepare the smoke test cases to check the readiness of the test environment setup.

|  |  |  |
| --- | --- | --- |
| **Entry Criteria** | **Activities** | Deliverable |
| Test plan is available.  Smoke test cases are available.  Test data is available. | Analyze the requirements and prepare the list of software & hardware required to set up test environment.  Setup the test environment.  Once the Test Environment is setup execute the smoke test cases to check the readiness of the test environment. | Test Environment will be ready with test data.  Result of smoke test case. |

**Test Execution:**

Once the preparation of test case development and test environment setup is complete. Then test execution phase can be kicked off. In this phase testing team starts executing test cases baes on prepared test Planning and prepared test cases in the prior setup.

Once the test case is passed then same can be marked as passed. If any test case is failed, then corresponding defect can be reported to developer team via bug tracking system & bug can be linked for corresponding test case for further analysis. Once the bug is fixed by development team the same test case can be executed.

If any of the test cases are blocked due to any defect than such test cases can be marked as blocked. So, we can get the report based on how many test cases passed, failed, blocked or not run e. Once the defects are fixed same failed or blocked test cases can be executed again to retest the functionality.

|  |  |  |
| --- | --- | --- |
| **Entry Criteria** | **Activities** | **Deliverable** |
| Test plan or test strategy document.  Test cases  Test data | Based on test planning execute the test cases.  Mark status of test cases like passed, failed, blocked, not run etc.  Assign bug ID for all failed and blocked test cases.  Do retesting once the defects are fixed.  Track the defects to closure. | Test case execution report.  Defect report |

**Test Cycle closure:**

Call out the testing team member meeting and evaluate cycle completion criteria based on test coverage, quality, cost time, critical business objectives and software. Discuss what all went good, which area needs to be improve & taking the lessons from current STLC as input to upcoming test cycles. Once the test cycle is completed then test closure report will be prepared.

|  |  |  |
| --- | --- | --- |
| **Entry Criteria** | Activities | Deliverable |
| Test case execution is completed  Test case execution report defect report | Evaluate cycle completion criteria based on Test coverage, Quality, Cost, Time, Critical Business Objectives, and Software  Prepare Test closure report  Share best practices for any similar projects in future | Test Closure report |

**Types of Testing**

* Unit Testing
* Black box testing
* White Box testing
* Grey box Testing
* System testing
* Functional Testing
* Integration Testing
* System and Integration Testing
* Regression Testing and Re- Testing
* user interface Testing (UI)
* Configuration Testing
* Back -End Testing
* Data Validation
* Data Migration Validation
* Data Conversion Validation
* Data Driven Testing
* Smoke Testing
* Sanity Testing
* Boundary Testing
* Positive Testing
* Negative Testing
* Parallel Testing
* UAT Testing
* Load Testing
* Alpha Testing
* Beta/ Acceptance Testing
* Graphical User Interface (GUI) Testing

|  |  |
| --- | --- |
| **Functional Testing Types Include:** | **Non- Functional Testing Types Include:** |
| * Unit testing * Integration Testing * System Testing * Sanity Testing * Smoke Testing * Interface Testing * Regression Testing * Beta/ Acceptance Testing | * Performance Testing * Load Testing * Stress Testing * Volume Testing * Security Testing * Compatibility Testing * Install Testing * Recovery Testing * Reliability Testing * Usability Testing * Compliance Testing * Localization Testing |

**Types of Testing**

**Unit Testing:** Testing of individual software components or modules. Typically done by programmers and not by testers. As it requires detailed knowledge of the internal program design and code, may require developing test driver modules or test harnesses.

**Black Box Testing:** Testing software based on output requirements and without any knowledge of the internal structure or coding in the program. Also known as behavioral testing. (“Also known as behavioral testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester.”)

**White Box Testing:** This testing is based on knowledge of the internal logic of an application codes and known as glass box testing, internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions. (“Also known as Clear box testing, Open box testing, Glass box testing, Transparent Box testing, Code based testing or Structural Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.)

**Grey Box Testing:** Testing software while already having some knowledge of its underlying code or logic, also defined as software testing technique which uses the combination of black box and white box for testing. (I a combination of white- box testing and black-box testing. The aim of this testing is to search for the defects if any due to improper structure or improper usage of applications)

**System Testing**: Is testing conducted on a computer, integrated system to evaluate the systems compliance with its specified requirements.  System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

**Functional Testing:** Functional Testing is part of system testing. testing software based on its functional requirements. It ensures that the program physically works the way it was intended, and all required menu options are present. In the functionality testing, we will concentrate only about the functionality in the application without bothering how it works in actual environment.

**Integration Testing:** Testing conducted to evaluate whether systems or components pass data and control correctly to each other.

**System and Integration Testing**: Is a testing process that exercises a software system’s coexistence with others. System Integration testing takes multiple integrated systems that have passed system testing as input and tests their required interactions following this process, the deliverable system is passed on to acceptance testing.

**Regression Testing and Re-Testing**: Is any type of software testing that seeks to uncover new errors or regressions, in existing functionality after changes have been made to the software, such as functional enhancements, patches or configuration changes.

**Configuration Testing:** Is the process of testing a system under development on machines which have various combinations of HW and SW.

**Back-End testing**: It is verifying how the application interacts with the database, interface communication, backend testing involves databases or any backend storage. It’s basically testing data while moving from Front to Back-End or in back end to back end only. The general list of things to be done are checking data integrity, data type, data size, performance, security etc.

**Data Validation Testing**: A process used to determine if data are inaccurate, incomplete, or unreasonable. The checking of data for correctness or compliance with applicable standards, rules, and conventions.

**Data Migration Validation:**  Is done to ensure that migrated data to storage type, format or system has been successfully and securely transferred (Validate all Migrated Data)

**Data Conversion Validation:** Is done to ensure that the converted data from one source to other been successfully and securely converted.

**Data Driven Testing**: Is the creation of test scripts to run together with their related data sets in a framework. The framework provides re-usable test logic to reduce maintenance and improve test coverage input and result (Test Criteria). Data values can be stored in one or more central data sources of databases, the actual format and organization can be implementation specific.  (Test Coverage is defined as matrix in software testing that measures the amount of testing performed by a set of tests)

**Smoke Testing**: Smoke Testing is general testing applied to test environments or application to provide a level of confidence that nothing has catastrophically failed, and that the most basic functions are available.

**Sanity Testing**: The aim of basic test is to be conducted whenever a new build is received for testing. Sanity testing is the subset of regression testing and it is performed when we do not have enough time for doing testing. Sanity testing is the surface level testing where QA engineer verifies that all the menus, functions, commands available in the product and project are working fine.

**Boundary Testing:** This testing technique involves inputting values at boundaries, just inside the boundaries, outside the boundaries and validating the output (Example MIN, MAX, AVG, MIN-1, MAX+1, decimal, negative, etc.)

**Positive Testing**: Positive Testing is that testing which attempts to show that a given module of an application does what is supposed to do. Positive testing = (Not showing error when not supposed to) + (showing error when supposed to)

**Negative Testing:** Negative testing is that testing which attempts to show that the module does not do anything that is not supposed to do. Negative Testing = (Showing error when not supposed to) + (not showing error when supposed to)

**Parallel Testing:** Running the same test on different systems to test behavior.

**UAT testing:** It is user acceptance testing. It is final step before deploying the application into production environment. Usually the end users who will be using the applications test the application before accepting the application. It gives the end users the confidence that the application being delivered to them meets their requirements.

**Stress Testing**: It is a type of non-functional testing. It is a form of software testing that is a form of software testing that is used to determine the stability of a given system. It put greater emphasis on robustness. Availability and error handling under a heavy load. Rather than on what would be considered correct behavior under normal circumstances.

**load testing:** Load testing is performed to determine a system’s behavior under both normal and anticipated peak load conditions. It helps to identify the maximum operating capacity of an application as well as any bottlenecks and determine which element is causing degradation.

**Alpha Testing:** It is the most common type of testing used in the software industry. The objective of this testing is to identify all possible issues or defects before releasing it into the market or to the user. It is the last testing done by the test teams after their acceptance testing and before releasing the software.

**Beta Testing-** It is a formal type of software testing which is carried out by the customer. It is performed in the real environment before releasing the product to the market for the actual end users. It sends the system/software to users who install it and use it under real-world working conditions.

**Graphical User Interface (GUI) Testing**

The objective of this GUI Testing is to validate the GUI as per the business requirement. The expected GUI of the application is mentioned in the detailed design document and GUI mockup screens.

The GUI testing includes the size of the button and input field present on the screen, alignment of all text, tables and content in the tables.

It is also validating the menu of the application, after selecting different menu and menu items, it validates that the page does not fluctuate, and the alignment remains same after hovering the mouse on the menu or sub-menu.

**Sequence of Testing**

Acceptance

System

Integration

Unit

**Environments for Testing**

Never test in a production environment. There are multiple instances and version of same software running in a company at the same time. Some of these may be:

* Production usually called “PROD"
* Test or Sandbox
* Development also called “DEV"

There may be numbers associated with them if they are more than one of the above.

**Test Documentation**

During testing the testing team develops various documents to support the testing process and reporting this includes:

* Test plan
* Test Case/Scripts (Test Scripts - Its a line by line description of all the actions and data needed to perform a test) (Test case generalize version)
* Requirement Traceability Matrix (RTM)
* Bug reports
* Testing Reports
* QA Sign off reports

**Difference between Smoke & Sanity Software Testing:**

Smoke testing is a wide approach where all areas of the software application are tested without getting into too deep. However, a sanity software testing is a narrow regression testing with a focus on one or a small set of areas of functionality of the software application.

The test cases for smoke testing of the software can be either manual or automated. However, a sanity test is generally without test scripts or test cases.

Smoke testing is done to ensure whether the main functions of the software application are working or not. During smoke testing of the software, we do not go into finer details. However, sanity testing is a cursory software testing type. It is done whenever a quick round of software testing can prove that the software application is functioning according to business / functional requirements.

Smoke testing of the software application is done to check whether the build can be accepted for through software testing. Sanity testing of the software is to ensure whether the requirements are met or not.

**Test Plan**

* A test plan is a document describing the scope, approach, objectives, resources, and schedule of a software testing effort. It identifies the items to be tested, items not be tested, who will do the testing, the test approach followed, what will be the pass/fail criteria, training needs for team, the testing schedule etc.
* A test plan is a detailed document that outlines the test strategy, testing objectives, resources (Manpower, software, hardware) required for testing, test schedule, test estimation and test deliverables.
* The test plan serves as a blueprint to conduct software testing activities as a defined process which is minutely monitored and controlled by the test manager.

**Importance of Test Plan**

* Test plan helps us determine the effort needed to validate the quality of the application under test
* Help people outside the test team such as developers, business managers, customers understand the details of testing.
* Test plan guides out thinking. It is like a rule book, which needs to be followed.
* Important aspects like test estimation, test scope, test strategy is documented in test plan, so it can be reviewed by, management team and re-used for other projects.

**How to write a Test Plan**

you already know that making a test plan is the most important task of test management process. Follow the seven steps below to create a test plan

* Analyze the product
* Design the Test Strategy
* Define the test criteria
* Define the test objectives
* Resource planning
* Plan test Environment
* Schedule & Estimation
* Determine test Deliverables

**Test Plan Template**

1. Table of Content
2. Objective/ Overview
3. Test Items
4. Scope (In scope-out of scope)
5. Test Strategy and approach
   1. Test Round
   2. Test Cases
   3. Test Resource
   4. Test Type
   5. Test Schedule
6. defect management and reporting
   1. Daily defect status report
   2. Weekly test status report
7. Traceability Matrix
8. Entry and Exit Criteria
9. Suspension criteria and resumption requirements
10. Test Deliverables
11. Testing tasks
12. Environmental needs
13. Responsibilities
14. Training needs
15. Risks and contingencies
16. Approvals

**Test Scenarios**

* A Test Scenario is any functionality that can be tested. It is also called test condition or test possibility. As a tester, you may put yourself in the end user’s shoes and figure out the real-world scenarios and use cases of the application under test.
* Scenario testing is a variant of software testing where scenarios are used for testing. Scenarios help in an easier way of testing of the more complicated systems.
* Scenario Testing is done to make sure that the end to end functioning of software is working fine or all the business process flows of the software are working fine. The
* testers take assistance from clients, stakeholders and developers to create test scenarios.

**Why create Test Scenarios?**

* Creating test scenarios ensures complete test coverage
* Test Scenarios can be approved by various stakeholders like business analyst, developers, customers to ensure the application under test is thoroughly tested. It ensures that the software is working for the most common use cases.
* They serve as a quick tool to determine the testing work effort and accordingly create a proposal for the client or organize the workforce.
* They help determine the most important end-to-end transactions or the real use of the software applications
* For studying the end- to - end functioning of the program, test Scenario is critical.

A screen shot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**Standards to follow to Develop Test Cases**

* Read specification and Requirements carefully
* Be clear with the design and the implementation details
* Analyze and identify all possible scenarios
* Identify the test environments and Test Types
* Should have detailed information related to affected areas of the requirement
* Should be clear of behavior under failure condition

**Testing Tools**

* HP ALM/Quality Center (Track Testing and Report Bug)
* TFS (Team Foundation Server) (Testing Management)
* JIRA (Track Testing and Report BUG)
* SELENIUM (Web Application Testing)
* QTP/ UFT (Automation Testing)
* Postman (API Testing)