VIRTUAL UNIVERSITY OF PAKISTAN

Software Quality Engineering

CS611

Course Handouts

GRADUP

Practice. Pass. Graduate.

thegradup.com

Software Quality Engineering

© Virtual University of Pakistan

Contents

Module-01: Software Quality Engineering Discipline 3

Module 02: Cost of Software Quality 9

Module 03: Standards and Models 13

Module 04: Engineering Process Area 20

Module 05: Process Management Process Area 33

Module 06: Software Requirement Engineering vs. Software Quality Engineering 41

Module 07: Quality Assurance Basics 51

Module 08: Software Quality Assurance & Defects 56

Module 09: Software Testing? 62

Module 10: Test Activities and Management 68

© Virtual University of Pakistan

Module-01: Software Quality Engineering Discipline

A. Quality Engineering Basics

What is software quality? What are the characteristics of high quality software solutions? What defines quality? These are some of the subjective question in the field of Software Quality Engineering. Modern Software Systems are usually interconnections of multiple underlying software and due to lack of standardization and varied nature it's really difficult to define quality. Software Quality Engineering involves complete software development process just to ensure that that any agreed-upon processes, standards and procedures are being followed to get desired results and there should be no cherry picking of standards

B. Roles and Responsibility

People may have different expectations related to software quality assurance based on their roles and responsibility. The stakeholders for software development are divided into two and their expectations are as follows:

C. Consumer

Consumers of a software product are further categorized into the following:

- ***Users*** are the group which use the services acquired by the customer: The quality expectations on the side of users are as follows:
    - It performs all the functions as specified in the software requirements, which fits/meets the user's needs.
    - Performs all the specified functions correctly over repeated use or over a long period of time, or performs its functions reliably.
- ***Customer*** usually acquire the Software and Services: The quality expectations on the side of consumer are as follows:
    - Basic expectations of the consumer are similar to that of users with additional concentration on the cost of the software solution.

D. Producer

Producer of the software solutions includes person involved in the development, management, maintenance and service of the software product. It also includes third party software product and organizations. For producers, the expectations are as follows:

- Their biggest concern is to fulfill their contractual obligations by producing software products that conform to product specifications.
- Proper choice of software methodologies, languages, tools, software usability and modifiability and other factors are closely related to quality for this category of stakeholders.

E. Off the Shelf Products

These are plug-and-play products and are usually known as Plugins. They are developed and tested independently of Software Solutions. Their main purpose is to provide reusable functionality. Off-the-shelf (OTS) software products can be defined as “software product(s) available for any user, at cost or not, and used without the need to conduct development activities”. Proper analysis should be perform while making decision regarding selection OTS as one solution does not fit all.

ISO-9126 Quality Framework

ISO-9126 is International Standard for Software Evaluation, it provides hierarchical framework for quality definition, organized into quality characteristics. There are six top-level quality characteristics that are summarized below:

F. Functionality

Functionality is the essential purpose of any product or service. The functionality characteristic allows drawing conclusions about how well software provides and performs desired functions. The functions are those that satisfy stated or implied needs. The more functions a product has, e.g. a sale order processing system, then the more complicated it becomes to define it's functionality. Continuing with the same example, the sales order system must be able to record sales, price, quantity, tax, shipping and inventory details. The software product may have multiple functions, but functionality is expressed as a totality of essential functions that the software product provides.

G. Reliability

The set of attributes related to the capability of software to maintain its level of performance under stated conditions for a stated period of time. The reliability characteristic tells the stakeholders about how effectively and efficiently a software solution maintains the level of performance if used under specified/stated conditions. Reliability can be used to evaluate the performance of whole or part of software and based on that suggest corrective measures to ensure continued software performance.

H. Usability

Usability can be defined as the ease to use any function especially from user view-point. Usability refers to the set of attributes of any software solution related to the individual assessment of different function by the stated users. The usability characteristic allows the stakeholders to conclude about how easily the solutions can be learned, understood and used. A good example to understand the concept is the revolutionary switch from Keyboard to touch-screen in 2007, and that makes Steve Jobs quote **“Machines can be user Friendly too”** a reality.

I. Efficiency

Efficiency is a set of attributes concerning with the relationship between the level of software performance and the amount of resources used, under stated conditions. This characteristic is concerned with the system resources (amount of disk space, memory, network etc.) used when providing the required functionality. This attribute examines how well the software provides required level of performance relative to the amount of resources used. For example, Good UI Design can take several minute to load due to bad internet connection and it may happen that Heavy weight UI might take more time to load in presence of good internet connection.

J. Maintainability

Maintainability refers to the set of attributes that bear on the effort needed to make specified modifications. In other words, the ability to identify and repair a fault within a software solution or any part of it is what the maintainability characteristic tackles. In simple words, the maintainability characteristic allows to conclude about how well software can be maintained. The analyzability, changeability, testability and stability are subcomponents of maintainability. This feature is easier said than done because it is directly related to how well or bad software is designed, documented and reviewed periodically.

K. Portability

Portability refers to the set of attributes related to the ability of software to be transferred from one environment to another. The portability characteristic tells about how well and easily software can be ported from one environment to another. Presence of functionality is required to measure. This attribute also refers to how well the software can adopt to changes in its requirements as well. Due to available to multiple platforms these days in 2017 this feature is very critical for the success because it might happen that one feature might work in one version of OS but fails to work properly in another version of OS of same platform

L. What is Error?

Error is a human action that produces an incorrect result and/or the mistakes made by programmer is known as an Error. Error is usually some syntax mistakes by developer but it can be both syntax and semantic error. This could happen because of the following reasons: some confusion in understanding the requirement of the software; some miscalculation of the values; or/and misinterpretation of any value, etc. Cost of fixing the logical error increases with line of codes to be analyzed.

M. Example of Error

Examine the following lines of code:

Semantic Error	Corrected Version

<?php
$Amount=100;
if ($Amount=100)
echo “Start calculation”;
Calculatetax();
else
Exit();?>
	<?php
	$Amount=100;
	if ($Amount==100) (Change the logic)
	echo “Start calculation”;
	Calculatetax();
	else
	Exit();?>

N. What is Defect?

Defect refers to the deviation from customer requirement. Mostly Defects are found in the Software after Software is shipped to the customer at production site. Defect is the departure of a quality characteristic from its specified value that results in a product not satisfying its normal usage requirements.

O. Example of Defect

Let’s assume a software solution for online payments. Following table would explain the user expectation vs. defect.

User Expectations	Software Defect
The software will allow me to make online payments using debit/credit cards	The option of selecting the debit card for making payments is missing in production Software

P. What is Bug?

Bugs are the errors found before the software is shipped into production. **Famously the defects accepted by developers are bugs** and software is shipped with known bugs. The ugly fact in the software development is that there is nothing like **Bug Free Software**. Most bugs results from mistakes and errors made in either a program's source code or its design, or in components and operating systems used by such programs. Bug is rarely traceable by Compiler to its nearest place.

Q. Example of Bug

July 28, 1962 -- Mariner I space probe. A bug in the flight software for the Mariner 1 causes the rocket to divert from its intended path on launch. Mission control destroys the rocket over the Atlantic Ocean. The investigation into the accident discovers that a formula written on paper in pencil was improperly transcribed into computer code, causing the computer to miscalculate the rocket's trajectory.

R. What is Fault?

An incorrect step, process, or data definition in a computer program is known as fault. Faults are fundamental condition within software that causes certain failure(s) to occur. Faults are known to be result of errors. In simple terms, Fault is an incorrect step or process due to which unanticipated result arises.

S. Example of Fault

Let's assume that the requirement is to write a program to add two numbers. In order to meet the requirement, the developer writes the following code:

```
#include<stdio.h>
int main ()
{
int value1, value2, ans;
Value1 = 5;
value2 = 3;
ans= value1 - value2;
printf("The addition of 5 + 3 = %d.", ans);
return 0;
}
```

Due to wrong sign there is deviation from expected result

T. What is Failure?

Failure is a result of fault; failure is inability of the program to behave as expected within given performance requirement. According to Laprie "a system failure occurs when the delivered service no longer complies with the specifications, the latter being an agreed description of the system's expected function and/or service". As mentioned above that failure is the result of fault, the following example would help understand this concept.

U. Example of Failure

```
#include<stdio.h>
int main ()
{
int value1, value2, ans;
Value1 = 5;
value2 = 3;
ans= value1 - value2;
```

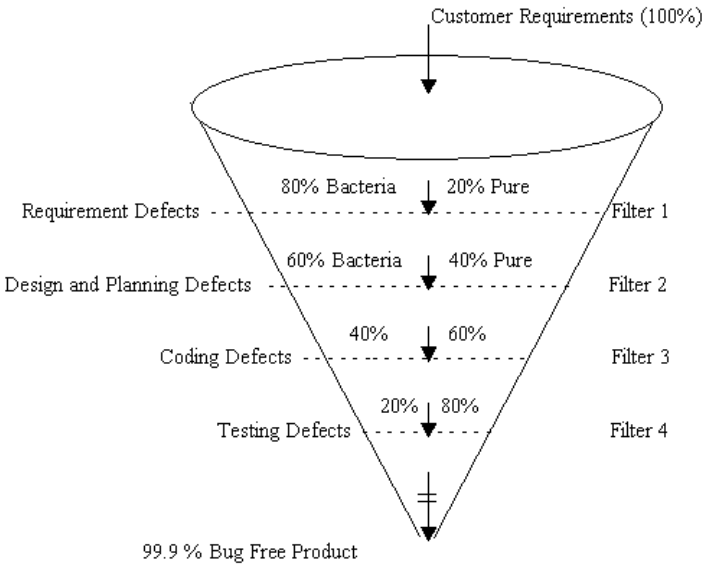
```
printf("The addition of 5 + 3 = %d.", ans);
return 0;
}
```

Fault: Due to wrong sign there is deviation from expected result

Failure: Due to Fault there is failure in the output. Instead of adding the two numbers it's subtracting the two numbers.

V. Defect Prevention

Recurring defects are very costly by nature and mere wastage of time and budget and on the same hand the challenge in any software product development lies in minimizing the number of defects. Defect Prevention is strategy to identify root causes of defect and prevent them from recurring. Defect prevention is one of the important activities in any software project. It is QA process to identify the root causes of defects and improve the process to avoid introducing defects, which help to improve the quality of the software product.



On a macro level defects can be classified and filtered as depicted in the figure. But still there is no bug free product i.e. 99.99% does not mean 100%

W. Defect Detection or Reduction

Defect Detection and Reduction is process to minimize defects but in a real scenario it is very **unrealistic to expect project or product with 0 bug count**. Defect prevention and defect reduction activities directly deal with the competing processes of defect injection and removal during the software development process (Humphrey, 1995). It is unrealistic to expect the defect prevention activities to be 100% effective in preventing accidental fault injections.

Therefore, we need effective techniques to remove as many of the injected faults as possible under project constraints.

X. Defect Removal or Containment

Due to nature of Software there are some defects which are produced under rare conditions. Defect Containment aims to reduce the chance of passing of defects from one phase to another. Due to large size and highly complex software systems, the defect reduction techniques only reduce the numbers of faults, though, to a very low level but this is not enough. The remaining faults may be triggered under certain and rare conditions. Thus it is necessary to prevent failures by breaking the causal relations between these faults and the resulting failures, thus "tolerating" these faults, or to contain the failures by reducing the resulting damage.

Module 02: Cost of Software Quality

Quality is always hard to define and in the case of software quality, it's more difficult. For any software application, the term quality may have different perception and definition among the developer, users, clients, managers, software quality engineers and other related stakeholders. Definition of quality often becomes even more complicated when quality depends upon the circumstances/environment in which it is being used. Literature reveals that software has the highest failure rate in the history of all the products resulting in loss of millions of dollars and this is one reason that makes quality important.

A. Economics of Software Quality Engineering

High concerns and challenges in the software quality engineering, one must realize the following facts in order to cope with the quality task:

- Everything in the process of software development ends up in the user's satisfaction
- Satisfaction of the user is dependent on the overall behavior of the system, and software product comes at first
- The behavior of any software product is defined and comprehended through features and quality
- Features and quality of the software product are defined/determined through **requirements**
- Any behavior related requirement of the software product can only be actualized through code that execute the behavior

Low software quality brings with it some serious economic consequences; therefore, it is important to know that only better than-average software quality has tangible economic values associated with it.

B. Function-Quality-Cost (FCQ)

The discussion on financial ramification of engineering quality into any software product can be summarized through the following statement:

In most development projects, functionality and quality (QA precisely) are natural enemies.

Projects with open budgets are very rare, usually the budget is fixed and here the functionality and quality compete with each other in order to get a bigger share from budget. The Function-Quality-Cost comes out to be:

= +

Where
A & B = Level of investment
F = Features/Functions
Q = Quality

It is very much clear that increasing feature in a closed-budget project will certainly decrease the budget share for quality of the product. The following example will elaborate the concept more clearly.

C. Quality vs. Pre-defined Budget

Let's take the example of project with fixed budget, say 100,000. Rest of the details would be as follows:

Quality vs. Pre-defined Budget Scenario	
Total Budget	PKR 100,000
Total Features	4
Cost per Feature	100,000/4 = PKR 25,000
Cost Breakup	
Development Cost of 4 Feature	PKR 80,000
Quality Cost of 4 Feature	PKR 20,000

In this scenario if the features are increased, there will be less budget for quality maintenance activities which is very natural and practical as companies are always under pressure to deliver more ignoring QA and its long term damage. Putting it theoretically, in a fixed priced budget project, the quality decreases if the number of features is increased in a closed budgeted project.

D. What are Missing Quality Requirements?

In a real-time scenario, more budgets mean more quality. This is both theoretically and practically true. Putting in more money for quality of the software product will result in low probability of product failure and may save a lot of financial resources as the high quality product will be immune to threats. For example, a software product with excellent User Interface (UI) but with no firewall for database security will face more threats. So adding a firewall to ensure Database is secure is more important than spending budget on cosmetic changes in UI.

E. Cost of Missing Quality Requirements

Lack of quality in any product can lead to massive losses but when we talk about lack of quality in software products, we can expect catastrophe. One such scenario occurred when Hackers access personal information associated with at least a half billion Yahoo accounts. This incident was report in **2016 but occurred sometime in late 2014**.

What was the ramification? Prior to the announcement of the breach, Verizon negotiated and decided to purchase Yahoo for $4.8 billion and this deal was to be closed in March 2017. But later in February 2017, Verizon and Yahoo announced that the deal will still go forward, but dropping the sale price by **$350 million and new offer was $4.48 Billion**. On the other side, user's confidential information including email, credit card details, bank account details and many others hit the market putting millions of users on stake.

F. Cost Analysis Based Approach

Missing Quality in Software Application has direct impact on People and Organizations as seen by the example mentioned in the above lines. Measuring such cost is critical to calculate impact and proceed with damage control otherwise the conditions will turn worst. Along with financial cost, there are other costs as well. According to **Eppler and Helfert principles** the costs are classified in two categories: direct and indirect.

G. Direct Cost of missing Quality

Direct Costs, as the name suggest, are directly linked to the missing quality. The direct costs are effects that are easily observable/measureable and they occur immediately after any unfortunate event. Examples includes; financial loss & physical injury and related. In short, direct costs are tangible, visible and measureable.

H. Indirect Cost of missing Quality

Indirect Costs are invisible cost of missing quality and hence difficult to calculate. It is also, sometime, difficult to realize or identify as they occur after a long time of the incident. Example includes: Loss of market share or reputation, loss of market and shareholders trust and

investment. Opposed to the direct cost, these are invisible as they may remain hidden for pretty long time, may have long-term impact as well. Scenario of Nokia serves a good example, its CEO said in May-2016 in his farewell speech: **"We didn't do anything wrong but somehow we Lost".**

I. Impact Analysis Approach

Missing quality attributes in software solution can impact both the customers and suppliers. The intensity or the impact of the loss may differ, but this thing is for sure that they'll bear some consequences. As in the case of Yahoo, the customers lost their privacy, their personal and business related confidential information. On the other side, Yahoo faced loss of trust; earn disrespect, financial loss, law suits and cost of investigation to find the root cause and others. Moreover, in certain situation customer may face cessation in business operation due to in process technical support or any kind of bug in the software solutions. In the worst case scenario, people are exposed to physical injuries to the extent of death. Impact analysis approach is based on the fact that one must perform

Root-cause analysis -> Identify problem->Fix it ->Keep Going because in fast paced world if you won't take appropriate action on right time then failure is inevitable

J. Risk Analysis Approach

Risk analysis approach is essential in determining the cost of missing quality. As in many cases, the time and place of missing quality events is difficult to determine, a better method of cost evaluation is risk analysis approach. The risk is defined by its probability (p) and its impact or potential loss (L). Risk exposure (RE) is the product of the risk probability and its potential loss. The equation could be:

$$(\quad) = (\quad) \times (\quad)$$

The probability and loss are directly and strongly related to the level of criticality of the software solution under observation. The different levels of risks are elaborated below.

K. Level of Risk

The IEEE Standard for Software Verification and Validation has published the most broadly known scale of criticality in the IT domain. The standardized IT system criticality levels are as follows:

- **Level A: Catastrophic**
  - Continuous usage (24 hours per day)
  - Irreversible environmental damages
  - Loss of human lives

    - Disastrous economic or social impact
- **Level B: Critical**
    - Continuous usage (version change interruptions)
    - Environmental damages
    - Serious threats to human lives
    - Permanent injury or severe illness
    - Important economic or social impact.
- **Level C: Marginal**
    - Continuous usage with fix interruption periods
    - Property damages
    - Minor injury or illness
    - Significant economic or social impact.
- **Level D: Negligible**
    - Time-to-time usage
    - Low property damages
    - No risks on human lives
    - Negligible economic or social impact.

Module 03: Standards and Models

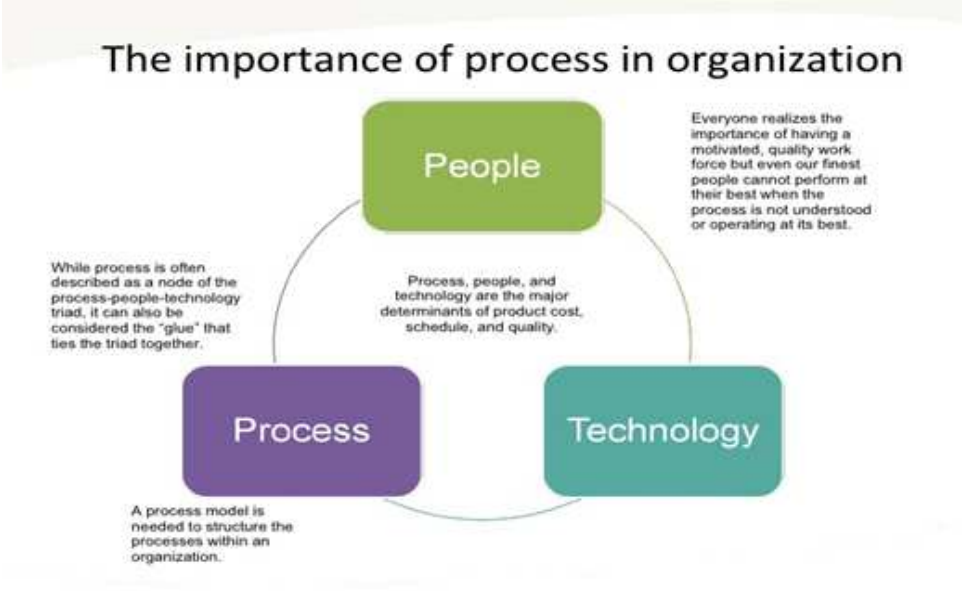
A. Rationale for Quality Management System

A quality management system is a formalized system to achieve Quality and the absence of which may lead to tragic situation or even product/system failure. A quality management system ensures documentation of Processes, Policies and work flows required to achieve desired standard of quality. One of the famous quality definitions - conformance to requirements - is a very unfortunate one because requirements are sometimes fill with defects, normally known as **toxic requirement**. It is for sure that conformance to those toxic requirements is not equivalent to quality. So the software engineering community has a moral obligation to eliminate such requirements.

B. Quality Leverage Points

One such framework to implement quality mindset is the concept of People - Process - Technology. It has also been referred to as the “golden triangle”. It reveals that finest Talent is

unable to perform due to lack of understanding of Processes and talent needs guidance to produce quality. That is why the Process part of People-Process-Technology triad is often called the leg of this triad. It works as a glue to keep together the other two aspects.



C. Why Process is needed?

Before the discussion of why a process is needed, let's understand process first. A process is a set of practices performed to achieve a given purpose more importantly practices are uniform and same across organization to perform a specific task. A process serves as an integration point which ensures synergy. Process doesn't work as a magic stick; it needs time to realize the results. Process provides a constructive, high-leverage focus on quality. The skills and training of the workforce is not always enough and working hard is not the optimal solution. A well-defined and implemented process can provide the means to work smarter, utilizing people and technology at optimal level. Technology, by itself, will most likely not be used effectively. Technology, in the context of an appropriate process roadmap, can provide the most benefit.

D. Process Benchmarking

Process benchmarking is a very important part of process improvement initiatives and it offers a variety of benefits including very critical and empirical data related to the organization's current processes and open room for improvements. Benchmarking is comparing existing processes and performance metrics to industry's best processes practices from other companies. But this should be kept in mind that there is not good or bad process; internal limitation, circumstances and resources must be evaluated before adopting any external process that seems to be optimal because what works in one situation might not work in others.

E. Organization vs. Processes

Organizations always need certain steps to achieve certain objectives. Those steps are carefully drafted and documented as Process along with some other critical elements like responsibility definition, process ownership, and process flow; in order to avoid any confusion. Processes, when implemented and followed correctly, ensure stability in results.

There was a time when processes were considered as overhead but with time and thanks to various researches, the processes are now considered as major enabler for organizational success. The focus on change management and organizational culture increased the relevance of processes in those areas. As mentioned above, there is no silver bullet to bring change in organization, change is best when it is slow, it requires consistency, vision and right sense of direction to bring change in organizations.

F. Mature vs. Immature Organization

Mature organizations are system oriented and they ensure stability. They rely on documented processes with clear sense of roles and responsibility at all levels. On the opposite side, immature organizations rely on gut feelings. Even if they have processes in place, they do not follow or implement them rigorously. Following table identifies major difference between mature and immature organizations:

Immature Organization	Mature Organization
Process improvised during project	Inter-group communication and coordination
Approved processes being ignored	Work accomplished according to plan
Reactive, not proactive	Practices consistent with processes
Unrealistic budget and schedule	Processes updated as necessary
Quality sacrificed for schedule	Well-defined roles/responsibilities
No objective measure of quality	Management formally commits

G. Process Model Overview of CMMI

Capability Maturity Model Integration (CMMI) is a collection or a model of best practices in systems, product and software development. CMMI is not a process and it does not tell how to do your work rather it tell what to do to achieve high quality. CMMI is based on the premise of Process Management. The CMMI provides a framework for organizing small steps into five maturity levels that lay successive foundations for continuous process improvement. The maturity levels have associated process areas. CMMI holds the following beliefs:

- Change should be normal and it must come slowly. Massive changes at once are doomed to failure
- Change should come in increments; in various steps.
- Change must come with future in mind, crisis prevention is better than recovering from crisis.

H. Behavior of Different Levels of CMMI

Each maturity level comes with set of best practices for implementation. When those best practices are implemented, each behavior is evaluated and appraised to measure its effectiveness. The results are compared with Metric (quantitative) based evaluation criteria which pre-defined for every behavior. Both the software process and products are quantitatively understood and controlled and the quantitative feedback enables continuous improvements. Further details of CMMI levels are given below.

I. CMMI Maturity Level 1 – Initial

At this level, the organization's environment is unstable for software development and maintenance. The processes - if any - well imperfectly defined and are reactive in nature. The organization, in overall is, unstable and unpredictable at this stage because the software process is constantly changed or modified as the work progresses. There is no roadmap for software development i.e., the process is ad hoc. Such organizations do face difficulties in retaining talented resources because of unstructured work and/or uncertainty in the organization. Let's examine a scenario:

J. Example CMMI Maturity Level 1

In a Software House, there are multiple projects in progress and projects can be assigned to single or multiple Project Manager, Assume there is new project which is assigned to two Project Managers, Client asks for what are next steps to proceed?

Answer:

PM-1: We will do Skype Call for team introduction

PM-2: We will send the Project Plan

Client: To whom I should believe!!!!!!

No responsibilities are defined, no roadmap for development. In other words, no process is in place whatsoever. There is uncertainty on both client and supplier side with no vision of future and the development process. Even if the company incorporates good software engineering practices, the benefits of those are undermined by ineffective planning.

K. CMMI Maturity Level 2 – Managed

At this stage, the policies and related frameworks are established for software development projects. Organizations at this level define a service strategy, create work plans, and monitor and control the work to ensure the service is delivered as planned. Besides work activities and processes are managed and ensured that they are planned in accordance with the policy. Organization defines responsibilities to avoid situation mentioned above and also provide adequate resources and training to the workforce so they can smoothly execute the process. This is still not the optimal stage as the process here are often reactive and organizations rely heavily on Heroes and when they are gone, process and performance are gone. Read the following scenario.

L. Example CMMI Maturity Level 2

A Software House which is Product Based which have around 400 + deployments at multiple client sites. At some point in time, client from Indonesia ask for estimates to develop new modules in the existing Product. Marketing team of Software House have meeting with Development Team to discuss requirement of new module requested by client.

After discussion there was a blocker issue **"One of the Software Architect"** was absent for last few weeks and there was no other resource that can help i.e. **that's what we call *HERO*** and add misery ***THERE WAS NO DOCUMENTATION* except the Software Architect himself, result is *COMPANY IS WAITING FOR HERO.***

Result: Client is shouting at Marketing Team and eventually stops using the Product

This results in supplier's credibility level going down to zero, leading to failure ultimately. Usually lack of documentation is justified with the intelligence and that's actually not true.

M. CMMI Maturity Level 3 - Defined

At the third Level, the standard process for developing and maintaining software are established and documented. The processes including both software engineering and management processes and they help workforce to perform more effectively. The reliance is on the defined process instead of Heroes. This stage can be considered as standard and consistent and people understand, support and follow the process and they are well aware of their roles and responsibilities. The major difference in Level 2 and 3 is as follows:

Level 2	Level 3
The process, standards and procedures are quite different for each instance of the process. The process can be different for a project or specific organizational unit.	The process, standards and procedures for a project are tailored from the organization's set of standard processes to suit a particular project or organizational unit.

N. Example CMMI Maturity Level 3

To-do List for Project Manager after Project Assignment

- Internal Kickoff to discuss and clarify scope related queries
- Client Kickoff
    - Team Introduction by Project Manager
    - Clarification of queries related to scope to be discussed and clarified from Client
    - Scope should be explicitly approved by Client to proceed to next step.
    - Meeting minutes to be shared with Client, Team Lead by Project Manager

In this scenario, a project manage have standard steps to proceed in order to successfully deliver the project.

O. CMMI Maturity Level 4 – Quantitatively Managed

At this level, Organizations quantitatively manage their process and software products. Quantitative objectives are established to evaluate the quality and process performance and hence they statistically analyzed. Management can measure different valuable metrics like software process, quality and productivity and they can also tune them as required. Quantitative boundaries are decided for the processes and organizations achieve control over their products and processes by narrowing the variation in their process performance to fall within acceptable range. During the evaluation, special variation points are identified for further improvements.

A critical distinction between maturity levels 3 and 4 is the predictability of process performance. At maturity level 4, the performance of processes is controlled using statistical and other quantitative techniques and predictions are based, in part, on a statistical analysis of fine-grained process data.

P. Example CMMI Maturity Level 4

Consider the following scenario:

In company where two stages of Project Kickoff are identified primarily Internal and Client Kickoff and participants are identified accordingly. Below is the time in hrs of the defined Process

Statistical Analysis of Kick-Off Process

Name	No of Participants	Planned Time (Minutes)	Total Man Time (Minutes)
Internal Kick-Off	4	60	240
Client Kickoff	5	60	300
		Total Time Spend on Kick-Off (Minutes)	540
		Total Time in Hours	9

Moral of the story is: ***"Process without Stats can't be improved"***

Q. CMMI Maturity Level 5 – Optimized

This is the optimal level where the focus is on continuous process improvement. The organization at this stage earns the ability to proactively evaluate the process in order to avoid the defects. Continuous process improvement is based on the quantitative understanding of the variation in the process performance. This level is all about striving for continuous improvements in the process capability and process performance. Such improvements occur in incremental changes in the existing process and by adopting new technologies and methods. The difference between level 4 and 5 is that:

- Level 4 focus on two things: addressing special causes of variation and providing statistical predictability of the results.
- Level 5 address common causes of variations and changing the process to improve performance and maintain the statistical predictability.

R. Example CMMI Maturity Level 5

Continuing with the example stated in level 4, there can be two options for further optimization. These are as follows:

Option-1:

Reduce Audience of Meeting

Statistical Analysis of Kick-Off Process			
Name	**No of Participants**	**Planned Time (Minutes)**	**Total Man Time (Minutes)**
Internal Kick-Off (PM, Team Lead)	2	60	120
Client Kickoff (Client, PM)	2	60	120
		Total Time Spend on Kick-Off (Minutes)	240
		Total Time in Hours	4

Option-2:

Merge Internal and Client Kickoff			
Statistical Analysis of Kick-Off Process			
Name	**No of Participants**	**Planned Time (Minutes)**	**Total Man Time (Minutes)**
Kick-Off (PM, Team Lead, Client)	3	60	180
		Total Time Spend on Kick-Off (Minutes)	180
		Total Time in Hours	3

S. Capability Level

Capability level is part of CMMI that is concerned with the capability of the organization relative to the process area. The capability level reflects on how well an organization is aligned to a specific process area. In CMMI there are different process areas and each process area have

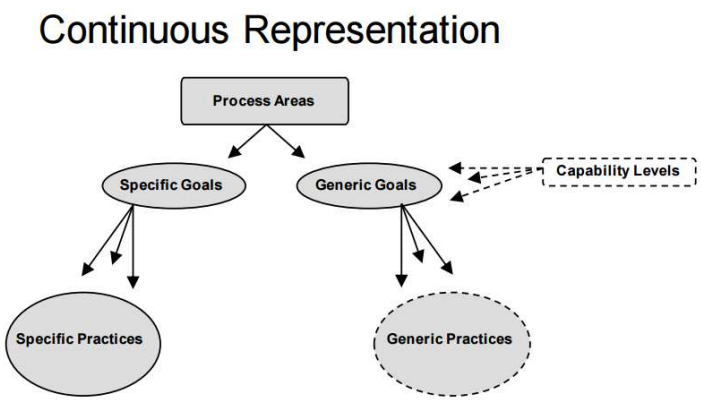
different processes. The capability level is consisting of specific and generic practices for a process area. Organizations can adopt those practices if they want to improve their processes associated with any process area. There are six capability levels designated by the numbers 0 through 5 and each level is a next step to the continuous improvement.

T. Component of CMMI Process Model

There are three components of CMMI process model through which maturity and capability are derived. These are as follows along with their actual definition and explanation as per CMMI:

- ***Process Area:*** A cluster of related practices in an area that, when implemented collectively, satisfies a set of goals considered important for making improvement in that area.
- ***Generic Practices:*** An expected model component that is considered important in achieving the associated generic goal. The generic practices associated with a generic goal describe the activities that are expected to result in achievement of the generic goal and contribute to the institutionalization of the processes associated with a process area.
- ***Specific Practice:*** An expected model component that is considered important in achieving the associated specific goal. The specific practices describe the activities expected to result in achievement of the specific goals of a process area.

U. Process Area, Goal and Practices



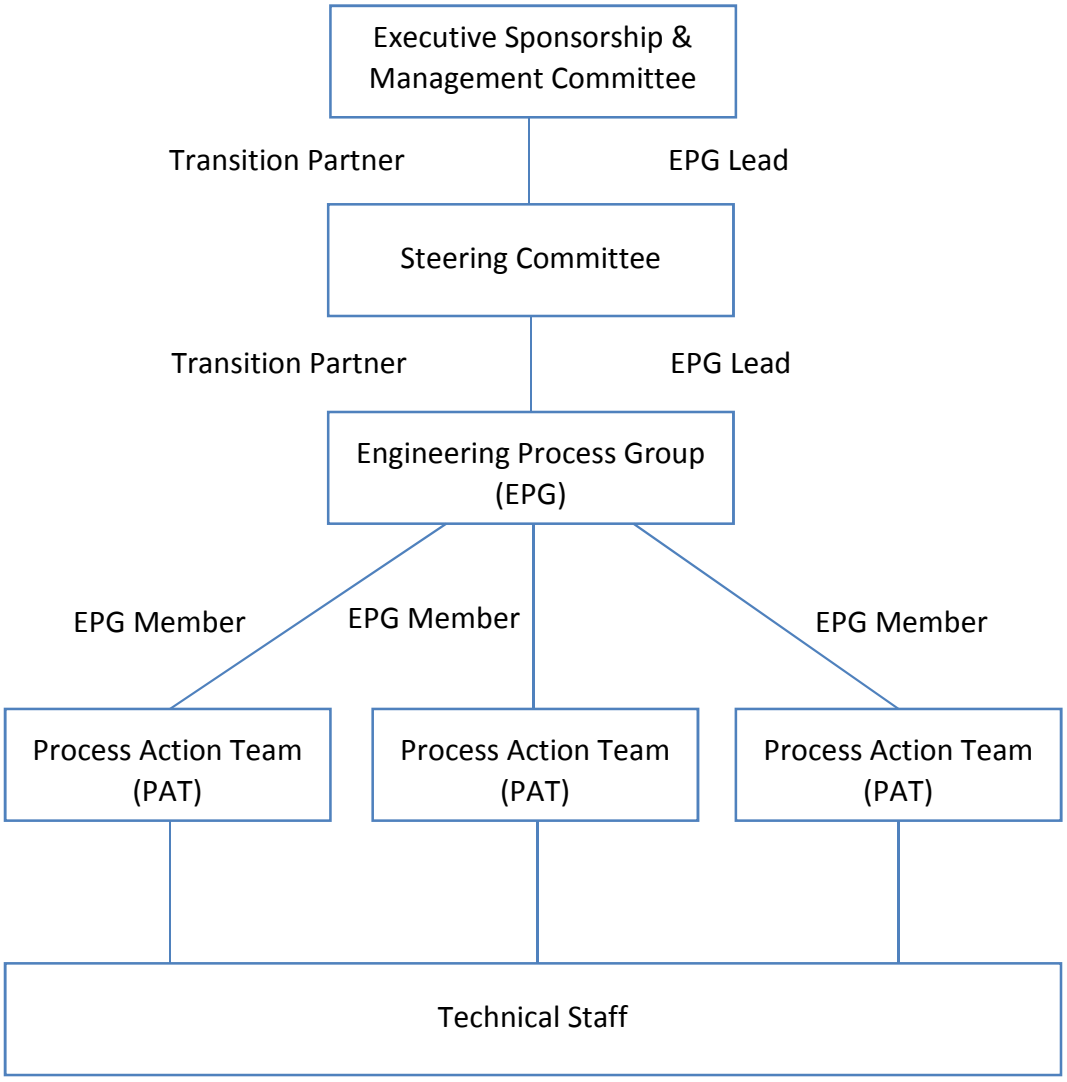
There are 24 process areas in total and each process area is associated with a maturity level. The optimal level in each process area is achieved in increments. Each process area has a set of standards, processes and guidelines that an organization must need to follow in order to achieve higher maturity level. Process areas are viewed differently in the two representations; continuous and staged.

- ***Continuous:*** the organization chose the processes that are critical to its business and achieve high capability levels.
- ***Staged:*** Organization using this approach achieve the goals of the process areas associated each maturity level.

Module 04: Engineering Process Area

A. Process Improvement Frameworks

The major purpose of engineering process group (EPG) is to improve the process throughout the organization. EPG first evaluate the existing process, define what a process should be and then provide suggestion for improvement. EPG also manage multiple process action teams with the purpose of improving different process areas simultaneously. PATs are individual teams created to address specific process improvement and PAT teams are consisting of technical staff from throughout the organizations. How EPG work is shown in the figure below followed by the details about process action teams.



B. Different Process Areas and Goal

There are in total 22 process areas but in this course only engineering related process areas will be discussed. As per CMMI definition, engineering process areas cover the development and maintenance activities that are shared across engineering disciplines. These are as follows:

- Requirement Management Process Area
- Requirement Development Process Area
- Technical Solution Process Area
- Product Integration Process Area
- Software Validation Process Area
- Software Verification Process Area

C. Process Action Teams (PAT)

As mentioned above, PAT is responsible for implementation of improvement initiatives activities in Specific Process Areas. In other words, each process area has associated process action team. The PATs are also known as the "worker bees." with immediate focus on weakness fund in process during the evaluation stage, their mandate is to write the procedures, pilot them, and update them as needed. Members of PAT belong to different domains and department of the organizations and they may include project managers. There tasks list is given below.

D. Task List of PAT

Process Action Teams (PAT) is mainly tasked to generate the process improvement documentation, policies, processes, procedures, charters, and Action Plans. For the improvement initiatives, PAT need to take care of different stakeholders for different process areas. One important task of PAT is to bring consistency in the documents throughout the organization in order to improve quality so they may need to work on drafting templates first. This will help in bringing same document structure for all processes and avoid rewriting of documents. This is also referred to standardization of artifacts.

E. Process Area: Requirement Management

This process area is concerned with the management of the entire requirement received or generated by the project, either technical or non-technical. The major purpose behind this is to ensure alignment between the requirements, project plans and the final output. One part of requirement management is to document the entire requirement, any changes in requirement along with their rationale. Change in requirements can take 2 forms, either change and/or update in the existing requirement or new requirement added to the project. Motivations behind requirement management process area are as follows:

- To manage inconsistencies between products and Requirements
- To manage different versions of Requirements

- To manage correlation between different project deliverable and requirements
- Traceability Matrix to be used to manage cross referencing

Action Item for Requirement Management

The goals of requirement management and practices to be followed are mentioned below:

Goal: Management Requirement

Practice: In order to achieve the goal, following practices are to be followed:

- ***Understanding Requirement:*** Develop an understanding with the requirements providers on the meaning of the requirements.
- ***Obtain Commitment to Requirements:*** Obtain commitment to requirements from project stakeholders. In other words, this specific practice deals with agreements and commitments among those who carry out activities necessary to implement requirements.
- ***Manage Requirements Changes:*** Manage changes to requirements as they evolve during the project using Change Management Process by performing Impact Analysis.
- ***Maintain Bidirectional Traceability of Requirements:*** When requirements are managed well, traceability can be established from a source requirement to its lower level requirements and from those lower level requirements back to their source requirements.
- ***Identify Inconsistencies:*** Ensure that project plans and work products remain aligned with requirements.

F. Example of Requirement Management

Requirement Management

Consider a real-time scenario below:

- 13-Mar-2016: Client and Project Manager agree on Requirements and Client approves it
- 14-Mar-2017: Requirements are passed on to Technical Team by Project Manager so they can work further
- 28-Mar-2017: Demo to be given to client and it was communicated to client
- 24-Mar-2017: Client and Project Manager agree on new set of requirements
- **28-Mar-2017: Client Reject the Demo by saying that Demo was not what was committed and rejected the Demo**

Root Cause Analysis

Client, Project Manager, Technical Team and QA were looking at different version of Requirements.

G. Process Area: Requirement Development

The purpose of this process area is to analyze and establish customer, product and product component requirements. Customer requirements are further divided into Product and Project Requirements. Requirements are identified and refined throughout the phases of the product lifecycle so all the requirements should be documented, analyzed and approved by the client and the source trace should be maintained.

Major artifact for this process area is Development of Software Requirement Specification (SRS).

Action Item for Requirement Management

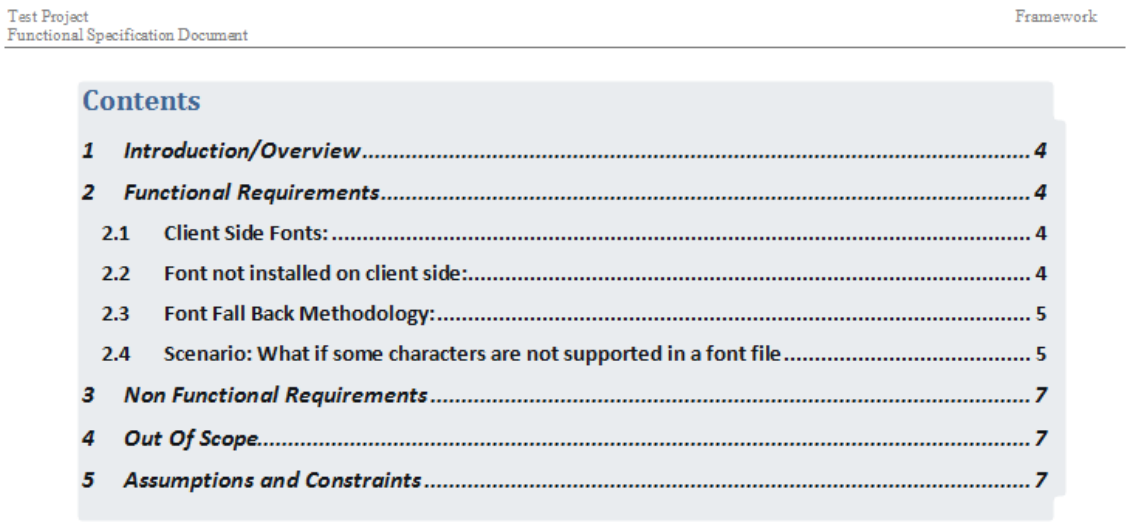
The goals of requirement development and practices to be followed are mentioned below:

- ***Develop Customer Requirements:*** Stakeholder needs, expectations, constraints, and interfaces are collected and translated into customer requirements.
- ***Develop Product Requirements***: Customer requirements are refined and elaborated to develop product and product component requirements.
- ***Analyze and Validate Requirements:*** The requirements are analyzed and validated.

H. Example of Requirement Development

The most important thing is that SRS should explicitly be approved by Client otherwise it will cause problem later in the Project.

The following images serve as good example of Requirement Development



Test Project
Functional Specification Document

Framework

Contents

1 Introduction/Overview 4
2 Functional Requirements 4
2.1 Client Side Fonts: 4
2.2 Font not installed on client side: 4
2.3 Font Fall Back Methodology: 5
2.4 Scenario: What if some characters are not supported in a font file 5
3 Non Functional Requirements 7
4 Out Of Scope 7
5 Assumptions and Constraints 7

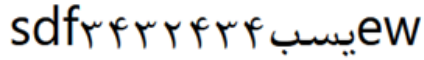
2.4 Scenario: What if some characters are not supported in a font file?

Consider a scenario, in which user has applied a Farsi or other font (for example "B Nazanin") which do not have English characters support. So what should happen, when user uses these fonts on IMAX resources like screens, Data Grid and Trends and the text contains some characters which do not fall under that applied font? Currently, square boxes are shown for the characters in the text if the characters do not fall under the applied font.

There are some proposed solutions:

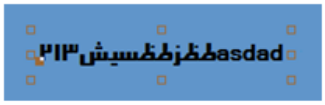
1. Live with the existing behavior i.e. square boxes will be shown for the characters which do not fall under the applied font. This is because the applied font does not have the support for those characters.
2. Tweak every font file itself to include the characters which do not fall under that font.
3. Try to implement the behavior which competitors are following:

ICONICS:

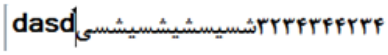


Here, "B Jalal" font is applied, which do not have the support for English characters. English fonts are rendered here with some default font.

Citect SCADA:
Text is placed on screen, and a Farsi font is applied on that text. All the characters are shown with respect to their fonts on screen as shown below:



Notepad:



Farsi font is applied to the whole text.

I. Process Area: Technical Solutions

This process area is all about selection, design and implementation of solutions to the requirement of the product/project. As a good practice, design and solution should be aligned and traceable with the requirements. In other words, the selected solution should produce the required output (requirement) and the solution must also tell that which requirement it is going to fulfill. This process area focuses on evaluating different solution or design approaches that satisfy functional and quality requirement. WireFrame are also considered, through partially, to visualize the technical design. The main goals for these process areas are as follows:

- ***Select Product Component Solutions:*** Product or product component solutions are selected from alternative solutions.
- ***Develop the Design:*** Product or product component designs are developed.
- ***Implement the Product Design:*** Product components, and associated support documentation, are implemented from their designs.

J. Example of Technical Solutions

The main artifact is Technical Design Document. Its purpose is to streamline the requirements, project plans and final output (product). All the details of adopted technical design are documented in this artifact which ultimately gives a picture of product architecture along with the traceability with the requirement. Sample Technical Design Document is attached in **Appendix - I**

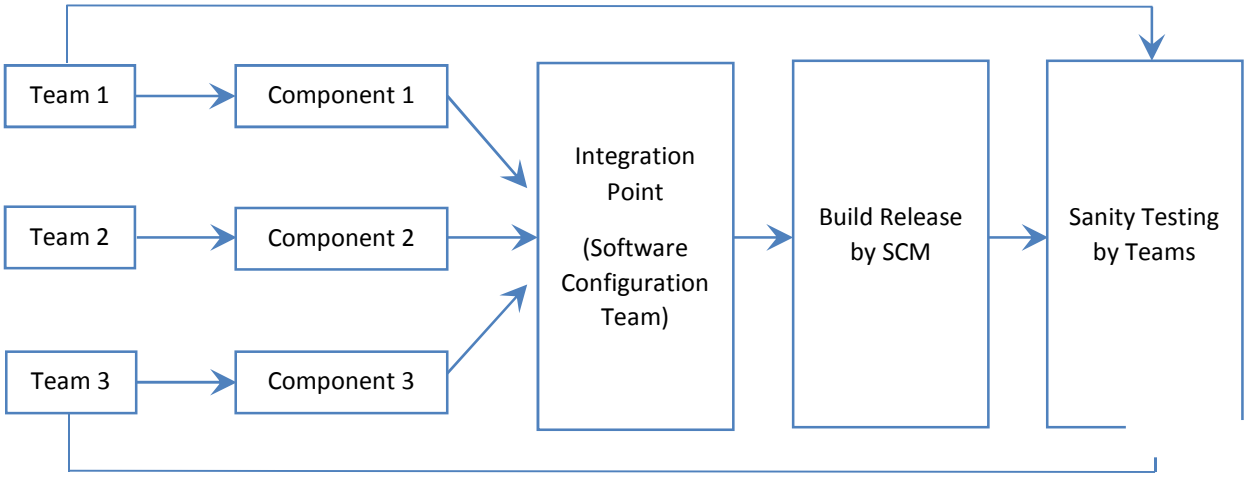
K. Process Area: Product Integration

Software products are made of different components and this process area is all about assembling the product from multiple product components and ensuring that the product (as a whole) behaves properly and satisfy all the functional and quality requirements. Major failure occurs when the product components are either failed to integrate with each other or partially integrate which results in defects due to misaligned interfaces. So, heterogeneous Development environment is a major risk in this area. Product integration is not one-time assembling of the product components; in fact it can be done incrementally. In other words, instead of simultaneous integration of all the components, only few components are integrated and tested first and then more components are assembled. Usually Sanity is performed to ensure that integration is successfully completed no further issues/defects are introduced due to it.

The main goals for this process area are:

- ***Prepare for Product Integration:*** Preparation for product integration is conducted.
- ***Ensure Interface Compatibility:*** The product component interfaces, both internal and external, are compatible.
- ***Assemble Product Components and Deliver the Product:*** Verified product components are assembled and the integrated, verified, and validated product is delivered.

L. Example of Product Integration



As per the example, there are three components developed by three different teams. These components are integrated by software configuration team and the merged code based is forwarded for sanity testing. Product integration also includes removal of issues on merged codebase. Sanity testing on merged codebase will evaluate the integration and check that no defects are introduced due to integration.

M. Process Area: Software Validation

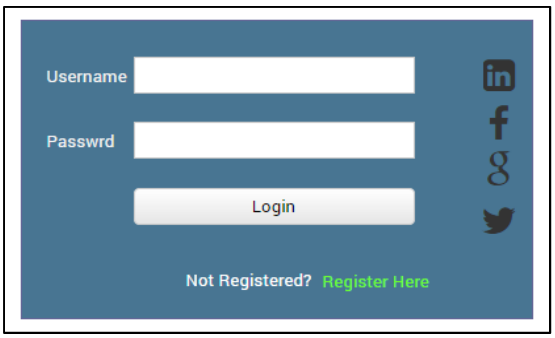
This process area has the purpose of ensuring that the final product or its component(s) fulfill the requirements and its intended use when deployed. The product or its components are validated in the intended environment be it manufacturing, operations or any other. The major goal is to capture client requirements correctly from client and then meeting that requirement i.e. building the right thing. No code is required for software validation as it is just to check whether the product is doing what it should be doing (as per requirements) in the intended environment. Once again, Proof of Concept, WireFrames, and Requirement Modeling are key to validation. The major goal for these process areas includes the following:

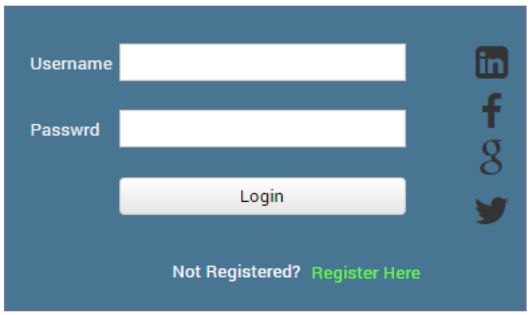
- ***Prepare for Validation:*** Preparation for validation is conducted by selecting the product, validation environment and validation criteria
- ***Validate Product or Product Components:*** The product or product components are validated to ensure they are suitable for use in their intended operating environment.

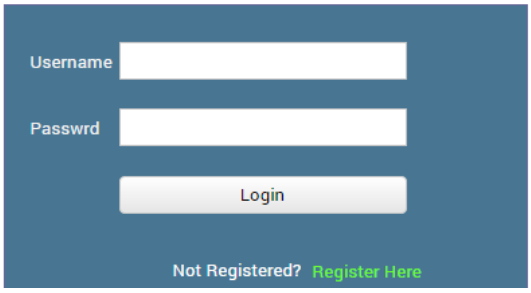
N. Example of Software Validation

Sample Requirements

- Admin (Employee) should be able to login
- Employee should be able to register another employee ¡n the organization
- Employee should be able to mark the attendance on daily basis







As per the requirement; employee is able to login, register another employee and mark attendance i.e. developer build the right product. One good strategy for software validation is that prototypes are shown to the customer and after approval the actual product is built and delivered.

O. Process Area: Software Verification

Software Verification process area is more concerned with the engineering/programming aspects of the project with the purpose to ensure that the final product is error free and selected work products/components meet their specified requirements. Verification does not

evaluate usefulness of the system instead verification is concerned with whether the system is well-engineered, error-free, and so on. So verification is more concerned with building the product right way. Software verification includes testing, design analysis, inspections and code reviews. The major goals for these process areas are as under:

- ***Prepare for Verification:*** Preparation for verification is conducted.
- ***Verify Selected Work Products:*** Selected work products are verified against their specified requirements.

P. Example of Software Verification

Continuing with the same example, QA team will verify that all requirements are being fulfilled and each part of the software is working properly. For example, in this case

Sample Requirements

- Admin (Employee) should be able to login
- Employee should be able to register another employee ¡n the organization
- Employee should be able to mark the attendance on daily basis

In verification QA team will execute each step after receiving the shipment from Development team

Q. Engineering Process Group

Engineering process group is organization's focal point to implement software processes to ensure compliance with quality standards. EPG also Act as oversight committee to monitor, evaluate and improve processes and it is a major player in coordinating process activity throughout the organization. Members of this group belongs to technical and management sides of the organization and they are responsible to assess the existing process, provide and implement suggestion for improvement and measure the effectiveness of the improved processes.

R. What are Audits?

Audit is tool to measure the organizational compliance level with the established process. Assessment and then improvements in any processes won't do any good for the organization if the process is not being followed. Through audits, organizations not only check the level of compliance but the reason behind the nonconformance. The reasons can come up in many shapes like people are not provided with required resources or training to follow the process or the process is misaligned with the working model and so on. Audits are conducted by independent auditor who first study the process, evaluate the conformance level and then assign ratings to the process after the audit.

S. Rationale for Audits

Audits are required to keep check and balance on organizational process and practices. Such audits become more important in volatile working environments. Results of audits become a starting point for process improvement and it gives valuable insights related to conformance. It can also tell where improvement is needed; in process design, process implementation, working conditions or the staff who are required to follow the process.

T. Audit Process

Figure 1: Continuous Audit Implementation Steps

U. Audit Types

There are three (3) types of audits and these, along with brief explanation, are as follows:

- ***First Party Audits:*** These are often described as internal audits. Someone from the organization itself audits a process to measure compliance and/or effectiveness.
- ***Second Party Audits:*** This is an external audit where the audit is being performed on supplier by a customer or by a contracted organization on behalf of a customer with the intention to ensure that the supplier is meeting contract specification.
- ***Third Party Audit:*** this is also an external audit and it's performed by an audit organization independent of supplier-customer relationship.

V. Audit Roles and Responsibilities

Audit team includes a certified lead auditor who leads the audit activities and a team of 2 to 3 members supporting the lead in performing audit. Each member is equipped with right attitude and skills to measure the process results and performing the process audit to ensure compliance. Further in tough scenarios, domain experts become a part of this team to deal with the technicalities of such scenarios.

W. CMMI Appraisals

Appraisal is defined as a process to collect, review and analyzes data to measure performance or compliance level. The collected data is then compared with the desired or standard data to identify the gap between the actual and desired, if any with the main purpose of measuring the effectiveness of the framework or process. CMMI appraisals provide ratings that accurately reflect the capability level or maturity level of the processes in use.

X. Process Reviews

Process reviews are frequently carried out in the organizations to measure the effectiveness of the process and to ensure that it is being completely followed. The frequency of process audit depends upon many factors but usually it's biannual or quarterly or on need basis. The process reviews also helps in identifying the required actions to improve the process results. Those required action may vary based on the results and it can be related to change/update in process objectives or design, training of stakeholders, technological advancement and many others.

Y. Review Policy

Every organization has its own review policy but the literature is full of best practices to be followed while conducting review. The fact remains same, the process is useless without review as no environment is static and no requirements are static. The current condition is best

described with the word dynamic or constantly changing and so must be the process to get desired results. Through review policy, organizations define what to review and when to review.

Z. Benchmarking for Process Review

There are many common issues the modern era organizations face today but the prospects of those issues or the circumstances might vary based on multiple factors. Benchmarking is the referencing to those common problems. For any issue, organization adopts the model of another organization that went through, more or less, same condition and developed a successful solution to it. Adopting the already developed solution after some amendments saves an organization a lot of time and efforts. Besides this, organization can use the best practices, with some modification, available in literature or research journals.

Module 05: Process Management Process Area

A. Need for Project Improvement (PM) Framework

Besides technical aspects, there are certain management facets of software development and these are managed with the frameworks of project management. Project management covers all the management related concerns of software development and sales. The main purpose is to standardize every step of software development life cycle (SDLC) and this can be achieved by developing company-wide consistent artifacts and frameworks. Instead of arbitrary software development, Project management suggests a systematic and consistent approach to be adopted throughout SDLC to ensure desired results.

B. Project Management (PM) Framework

Project Management Framework is a bridge between Development, Sales, Finance and Management. A project contains handsome number of factors contributing for desired results and none of them can be put in isolation as it may harm the project. Project management is an approach to connect technical and management facets of software development so the product is delivered with the desired quality and time span while reducing costs. The project management involves project planning and execution of plans, management of software development teams, project documentation and project monitoring. Auditable data is incorporated in all phases of project management.

C. Component of Project Management Framework

The following document sample would help a lot to get a fair and practical idea about the project management framework.

Project Management Framework

a. **Objectives:** To ensure visibility of Project Progress to be Audited any time during Project lifecycle
b. **Audience:** Higher Management
c. Project Name:

Following are the guidelines to be followed during complete lifecycle of the Project execution after contract is signed.

d. Scope Guidelines:

Requirement clarification and Scope management can be clear in phase-wise manner or completely at once, it will depend on Project.

i. Discussed and baseline with Sales: Invites are to be created by PM (Sales Rep, PM, TL, Designer, VP-Delivery).
ii. Discussed and baseline with Customer (if needed): Invites to be created by PM(CEO, Client, Sales Rep, PM, TL VP-Delivery)

Phase	Approved by Sales	Approved by the client	Link to the approvals
1 (complete)	✔ Sep 18 2014	✔Sep 17 2014	

e. Budgeted effort

Phase	Effort as per contract	Actual Effort (as per estimates)	Variation above 25%	Meeting with sales on higher variation (more than 25%)	Link of Discussion
1	40	40	No	NA	
2	40	75	Yes	✔ Oct 2 @ 12:00 Pak Time	
3	72	134	Yes	✔ Oct 2 @ 12:00 Pak Time	
4	48	48	No	NA	

f. Roadmap

Phase	Dev Start Date	Dev End Date	Invoice Date
1	29-Sep-14	14-Oct-14	17 Oct ✔Sent
2	29-Sep-14	21-Oct-14	28 Oct ✔Sent
3	14-Oct-14	17-Nov-14	⌛ 20 Nov
4	17-Nov-14	24-Nov-14	⌛ 26 Nov
Note: Invoice Deadlines are to be shared with Finance Dept.			

g. Demo plan

Demos are to be given by PM or in case PM is not available or on leaves his replacement will do the demo

i. Invites for Internal Demo (Sales Rep, VP - Technology, VP-Delivery, PM, TL, Designer):
ii. Invites for External Demo (CEO, Client VP - Technology, Sales Rep, VP-Delivery, concern PM, TL)

Phase	Internal Demo	Status	External Demo	Status	Link of MOM (AC)
1	15 Oct @15:00 PKT	✔MOM	16 Oct @ 15:00 PKT	✔MOM	
2	22 Oct @15:00 PKT	✔MOM	24 Oct @15:00 PKT	✔MOM	
3	17 Nov @ 13:00 PKT	⌛	18 Nov @17:30 PKT	⌛	
4	Demo if needed	⌛	Demo if needed	⌛	

h. Updates plan

In case there is deviation from plan or if a deadline is missed following steps are to be done:

iii. Updated Plan to be shared by PM
iv. Updated deadline to be shared with Client and Finance by PM

Note: For Invite detail Date / time is needed to be mentioned.

D. Artifacts of Project Management Framework

There are following artifacts of PM Framework:

- Signed Contract
- Specification Documents
- Estimations
- Gap Analysis
- Project Plan
- Demo Plan
- Invoice Plan

E. Project Planning

Project planning includes a wide range of activities including development efforts, quality assurance and demo dates. Planning for development activities requires estimation containing man-hour (and budgetary) requirement to complete the development. This estimation is prerequisite to the project planning. A point to be noted here is; the estimations are totally different from timelines. 200 hours does not mean 5 weeks. The estimation is usually worked out in terms of man-hours while timeline is progress of activities on a calendar timescale.

F. Example of Project Planning

Start Date	1/2/2017
End Date	15/4/2017

No.	Task	Estimate in hours	Start Date	End Date	Assigned To	Status	Comments
1	**Add sorting filters all over the listings**						
	- Design work	12	1-Feb-17	2-Feb-17		Completed	
2	**Deactivated users module**						
	- add tab for Deactivated user	4	2-Feb-17	2-Feb-17			
	- list for the deactivated users	8	3-Feb-17	4-Feb-17		Completed	

	- Add time filters (date only)	6	4-Feb-17	5-Feb-17		Completed	- assuming that we don't have any deletion and edit function
3	**Analytical report time filters**					Completed	
	- add drop down	2	5-Feb-17	6-Feb-17		Completed	
	- handle change listners on all dropdowns	5	6-Feb-17	6-Feb-17		Completed	
	- Average karma points spent By users	6	7-Feb-17	7-Feb-17		Completed	
	- Total karma points given to projects	3				Completed	
	- Total karma points given to participaid	3				Completed	
	- Total karma points spent on wishes	12				Completed	
	- Total karma points spent on offers	3				Completed	
	- Total karma points earned on sign up	8				Completed	
	- Total karma points earned on needs	24				WIP	
	- Total karma points earned on inviting	3				WIP	
4	**Participaid Profile page (CMS)**						
	- Participaid link on menu	3				Pending	
	- Design integration / template design	8				Pending	
	- add slider on front end	16				Pending	
	- Admin side option in menu	2				Pending	
	- admin side Form to add image text and link	12				Pending	
	- News widget for frontend	6				WIP	
	- Admin side form to add title detail image and link	48				Pending	
	- show total number of active users of participaid	4				Pending	
	- show total number of active projects of participaid	4				WIP	
5	**Hide participaid profile page**	5				Pending	
6	**Design change and addition**	48					
7	**QA**						
8	**Internal Demo**						

9	**External Demo**						
	Total Hours	**255**					

Project Management and Testing	48
Total Hours	255 + 48 = 303
Total Man Days	**303 ÷ 8 = 37.875**

G. Project Tracking and Control

The purpose of tracking the project is to ensure project is moving ahead as per the plan. Daily standup meetings (SCRUM Meetings) are planned to maintain daily progress and to improve communication among team members. Besides weekly status meeting are key to execute projects as per planning. Again, project planning is prerequisite to project monitoring and weekly status report is published to document the project status and updated all team members.

H. Example of Project Tracking and Control

Following is a sample weekly status report showing the planned activities along with completion status.

Sample Weekly Status Report

Project Name	1880
Planned Start Date (As per contract)	2-Jul
Planned End Date(As per contract)	2-Oct
Actual Start Date:	25-Jul
Estimated End Date (if different from planned end date)	14th November
Planned % Completion	90%
Current Project Completion Status (Mention % complete)	4%

Milestone# / Phase	Planned Start Date	Actual Start Date	Planned End Date	Estimated End Date (if different from planned end date)	% Completion (if Milestone not completed)	Comments
Phase 1	2-Jul	2-Jul	8-Jul	26-Aug	100%	
Phase 2	9-Jul	18-Aug	15-Jul	27-Aug	100%	
Phase 3	16-Jul	27-Aug	25-Jul	19-Sep	80%	

Depending on Nature of Project Multiple Status Reports can be generated.

I. Audit of Each Phase of PM Process Area

To determine the project health, project managers perform Project Audit which covers all the process areas of project management. Project Manager is responsible to provide relevant data for Audit. The purpose is to evaluate the extent to which project management standards are being followed throughout the project. Further it also assesses the project quality and the reason to the known problems of the whole project and helps in taking corrective measures. Through this report, whole take get a clear summary of individual phases of the project. Here is a summarized example of project audit:

Sample Project Audit Report

Project Name	A	B	C	D	D	E	F	G
Scope	yes	No	Yes	Yes	Yes	Yes	Yes	No
Budget	yes	No	Yes	Yes	Yes	Yes	Yes	No
Roadmap	yes	No	Yes	Yes	Yes	Yes	No	No
Demo Plan	yes	No	Yes	No	No	No	No	No
Updated Plan	NA	NA	No	NA	NA	Yes	No	No
Project Health	Good	Critically Deficient	Good	Satisfactory	Satisfactory	Good	Weak	Critically Deficient

Response Glossary	
Critically Deficient:	Serious inability to comply with the expectation.
Weak:	Unable to entirely comply with the expectation.
Satisfactory:	Basic objectives met but room for improvement.
Good:	Exactly as per the expectations.
Very Good:	Exceeded the expectation. Role model for other

The actual project audit report is detailed and includes real-time data for all the phases and finally suggests the improvement required.

J. Earned Value Management (EVM)

EVM is a tool to provide objectives measures of cost and schedule performance of the whole project. The purpose is to timely highlighting the cost and schedule issues so that the remedial actions are taken before it's too late to recover. Earned value is expressed as a budgetary value

of the work done on a project but still it's a project management tool rather a financial tool. The results are presented in a graphical format. An example is hereunder:

Three values are being evaluated here:

- Planned Value (PV)
- Actual Value (AV)
- Earned Value(EV)

K. SPI and CPI

Schedule Performance Index (SPI) is used to determine project schedule and it is expressed as the ratio of earned value and planned value. Following equation is used to do so:

$$= \quad (\quad) \div \quad (\quad)$$

The formula concludes as follows:

- If the SPI is greater than 1, it means the project is ahead of schedule.
- If the SPI is less than 1, it means the project is behind schedule.
- If the SPI is equal to one, it means the project is on time

Cost Performance Index (CPI) is used to determine project budget status and it is expressed as the ratio of earned value and actual value. Following equation is used to do so:

$$= \quad (\quad) \div \quad (\quad)$$

The formula concludes as follows:

- If the CPI is less than 1, it means the project is over budget.
- If the CPI is greater than 1, it means the project is under budget.
- If the CPI is equal to one, it means the project is as per the planned budget.

L. Example 1 of CPI and SPI

Consider the following example. As per the result Project is over budget and behind schedule

Example 1 of CPI and SPI	
Project Audit - 20-April-2017	
Planned Value (PV) - Planned Effort (Contracted Effort)	206
Actual Value (AV) = Actual Effort from Weekly Status Reports	358*
EVM: Completed Tasks Effort as of Project Plan All Task marked completed in Plan after 6-Week	230**
CPI (EVM/AV)	0.642458101
SPI (EVM/PV)	1.116504854
Result	Over Budget --- Behind Schedule
Over Budget	Actual is greater than what is earned
Behind Schedule	Earned is less than Planned

* Sum of Actual Effort Planned to be completed by 20-April-2016

** Sum of all tasks with Status = Completed by 20-April-2016

M. Example 2 of CPI and SPI

Consider the following example. As per the result Project is over budget and behind schedule

Example 2 of CPI and SPI	
Project Audit - 20-May-2017	
Planned Value (PV) -Planned Effort(Contracted Effort)	112
Actual Value (AV)= Actual Effort from Weekly Status Reports	216*
EVM: Completed Tasks Effort as of Project Plan All Task marked completed in Plan after 6-Week	40**
CPI (EVM/AV)	0.185185185
SPI (EVM/PV)	0.357142857
Result	Over Budget --- Behind Schedule
Over Budget	Actual is greater than what is earned
Behind Schedule	Earned is less than Planned

* Sum of Actual Effort Planned to be completed by 20-May-2016

** Sum of all tasks with Status = Completed by 20-May-2016

Module 06: Software Requirement Engineering vs. Software Quality Engineering

A. Cavendish Software Chaos Report

According to the Cavendish software chaos report, the success rate of software and IT project is getting better in the recent years but still failure rate touches a high percentage and this is a major point of concern for the industry professional. The mentioned report state three major reason behind the failure and requirement engineering is on top with 87% of failure chances if not done accurately. Requirement engineering poses a big challenge for the industry as doing it properly is strenuous task in today's dynamic and turbulent environment.

B. Motivation for Software Requirements

As mentioned above, requirement engineering is easier said than done and become harder task when client is not mindful of the requirement and put requirement documentation on low priority. This leads to ambiguity and uncertainty throughout the project and the chances of failure become so high because the development team is not completely sure about what is to

be done. In the absence of proper documented requirements, the actual purpose of software development is lost.

C. Why to Focus on Requirements?

Requirement engineering is the most important area of the entire software life cycle because error produced in this phase cost a lot in later stages and customer will not get the required product in the required time. Again, as mentioned above, faulty or missing requirements are the biggest reason behind the project failure. The following problems can occur during requirement engineering:

- ***Missing Requirements:*** if not documented properly, project team may skip requirement(s) and it would be nearly impossible to change the software design afterwards. And sometimes, most critical requirements are missed.
- ***Client not Involvement:*** due to client's lack of involvement, requirements can be misunderstood or miscommunicated leading to faulty production.
- ***Wrong Requirement:*** Due to less technical knowledge at client's end, the requirements are sometimes technically wrong or they are in conflict with some other requirements.
- ***Changing Requirements:*** If the client is not completely informed of the requirements, he may make changes in later stages of development that might become difficult to incorporate.
- ***Out of Scope Requirements:*** The requirements stated by client may go out of scope of the project.

D. Effort-wise Distribution of SDLC

A requirement document is considered to be an overhead in SDLC and as shown in the graph, it takes only 10% of the whole project budget. Ideally, there should be 20% budget for requirement gathering and documentation. Because negligence in this phase will negatively affect the entire project in terms of quality, cost/budget, deliverables, meeting customer requirement and revenue.

E. Requirement Defined

A requirement can be defined as a feature of the system or a description of something the system is capable of doing in order to fulfill the system's purpose. A point worth understanding is that the requirement engineering in not a technical activity rather it is purely a communicational activity. Different stakeholders from all the facets of software development may have different concerns and point of views. But the challenge is to create a win/win situation for all stakeholders. Another impediment is to understand the requirement from a nontechnical client because they are difficult to handle because of knowledge issues.

F. Software Requirements vs. Requirements

Software requirements are not similar to the requirement in other domains. Software requirements comes with a lot of variation and changes (in all stages of SDLC) as compared to other domains where requirements are completely understood and defined in the beginning of the project and they remain same throughout. Software product is not tangible until the first demo and requirements changes after it. For example, in the construction industry the requirement and design are completely understood and they remain fixed throughout the project. But that's not the case with software development where requirements changes during all stages of SDLC.

G. Attributes of Software Requirements

There are multiple attributes of software requirements and they are explained with examples in the following headings.

H. Attribute 1: Correct

Requirement should capture the client expectations and there is not be any kind of ambiguity in that. All stakeholders must be communicating openly to correctly analyze the

expectations/needs of client and converting them into software requirement. As a good practice, requirement should be stated in single line and must not exceed 30 to 50 words in length. Open ended and subjective terms should be avoided as they may cause misunderstandings. Terms like 'etc.' or 'assumed' are totally not appropriate for documenting requirement.

I. Example of Correctness

Requirement: GUI should be user friendly	
Issue	**Suggestion**
What is user friendliness? Is use of more images or text is desirable or is there any preferable color scheme?	This sort of requirements is very difficult to be correct in first cut. Ask client for reference or show him some reference material or sample website to get start with. ***Avoid falling in the trap of using subjective terms.***

J. Attribute 2: Coherent

Requirements must be coherent, consistent and they must not come in conflict either with other requirements or the project scope. While documenting requirements, vocabulary or any technical terms mentioned in the statement must be consistent and defined so that all stakeholders are on same page. The requirement should be logical and it should add value in the project.

K. Example of Coherent

Requirement 1	***Requirement 2***
Customer support should be IT ISO-ITIL 3.0 compliant	The network support should be ISO-90 – ITIL 3.0 compliant
Question: Which one is correct version of ISO or both are differently correct? *Answer: Only one standard should be used.*	

L. Attribute 3: Complete

For software requirement engineering, an assumption is the mother of all failures. The requirement should be complete so all stakeholders completely understand them it must not leave any room for guessing and assuming things. To avoid ambiguity, a requirement must express the entire need and state all conditions and constraints under which it applies.

Incomplete requirements can lead to faulty system development and ultimately customer will reject it and users will be unsatisfied.

M. Example of Completeness

Requirement: The product shall provide status messages at regular intervals not less than every 60 seconds.	
Issue	**Suggestion**
Requirement is incomplete Open Question: What are status messages and where are they supposed to be displayed?	Status messages to be displayed on designated areas on UI or they should come as Pop-up messages. ***Requirements should be closed ended as much as possible***

N. Attribute 4: Feasible

A requirement is considered to be feasible if it is possible to implement it within the scope and limitation of the project. Or in other words, a requirement would be feasible if it can be satisfied. The best practice is to involve and engage developer(s) to provide technical insights about the requirement. It has been said that having zero bug in a software product is not a feasible requirement as it may take a long time to prove it.

O. Example of Feasible

Requirement: The product shall switch between displaying and hiding non-printing characters simultaneously	
Issue	**Suggestion**
Not Feasible Reason: Computers cannot do anything simultaneously, so this requirement is not feasible	The user shall be able to toggle between displaying and hiding all HTML tags and non-HTML tags ***It is very difficult to identify feasible or non-feasible requirement without technical input.***

P. Attribute 5: Necessary

A requirement must be necessary for the software system and should be adding some business value. This type of requirement comes from the business goals and the need behind developing the software solution. In the absence of such requirement, the system would not be able to function properly as per the need of the client. So the project team must work on tasks that fulfill client needs instead of working on what they want.

Q. Example of Necessary

Requirement: Clients want login integration via social media (any one) beside normal sign-in	
Issue	**Suggestion**
Developer become excited when implementing login via social media and implemented 3 different login via 3-different social media applications and forget normal sign-in. Client totally reject the demo as for Client normal sign-in have more Business value from customer view point	Always follow customer requirements or whatever fulfills the customer's needs. ***Focus should be on what is expected by client from market view-point.***

R. Attribute 6: Verifiable

A requirement can be considered as verifiable if it can be implemented in the system and so it is demo-able. The quality assurance team should be able to close the requirement after verification. Requirements which are not testable are not verifiable requirements. As a good practice, each requirement must be expressed unambiguously to make it a verifiable requirements and words like 'must', 'shall' or 'etc.' must be avoided while documenting the requirements.

S. Example of Verifiable

Requirement: Loading time of the website should be as minimum as possible?	
Issue	**Suggestion**
What is minimum? Minimum is not defined and hence QA can't test it i-e what is value for which QA should test, result there will always be conflict between development	Always ask for numeric value, Minimum loading time should be at most **200 milliseconds** *Always look for keywords to be avoided and*

and QA team.	*get requirements vet from QA department also.*

T. Attribute 7: Traceable

Being traceable, requirements should be linked to source in Requirement Specifications. A requirement will be considered as unique if it has a unique identifier like some ID number. The traceability feature of requirement enables to ensure the product is built as per the need/requirements. As linked with the source, the requirements must be linked with Use cases, design and test cases. Requirements should have forward and backward integration. Forward integration is used to identify where does this requirement get used while backward integration is used to know where did this feature originated.

U. Example of Traceability

Consider the following scenario:

After Project fail to deliver on time, process of root-cause analysis (RCA) was initiated to identify reasons due to which team is unable to deliver on time. Core reason which comes out of RCA that audit team is unable to identify requirements on which key developer was working and developer spend major part of his time on this non-traceable requirement. In Specs or SRS there was no discussion on a feature on which time was spent.

Non-traceable requirements often lead to Project failure – This RCA is from real-time analysis, intentionally names of project is omitted

V. Functional Requirements

Functional requirements define what system should do and how the system should behave under certain situation or in a specific environment. In other words, the functional requirements describe the core functionality of the application. Functional requirements drive the application architecture of a system may include calculations, technical details, and other specific functionality that define what a system is supposed to accomplish. Functional requirements are captured using Use Cases (a use case is a list of actions or event steps, typically defining the interactions between systems, to achieve a goal.)

W. Template for Functional Requirements

Following template can be used to document functional requirements:

Sample Template for Capturing Functional Requirement

Use Case Name:*	
Use Case ID	
Actor:	
Summary:	
Pre-Condition:	
Post-Condition:	
Extend:	
Uses:	
Normal Course of Events:	
Alternative Path:	
Exception:	

** Use cases serve as basis as connector between different phases of SDLC*

X. Example - 1: Filled Functional Requirements

Use Case Name:	Sign In
Use Case ID*	1
Actor:	Administrator, Super User, Doctor, Receptionist. Accountant, Lab Manager
Summary:	This use case describes the scenario in which actor logs into the system prior to its usage.
Pre-Condition:	Actor must be already registered and authorized with the system.
Post-Condition:	If the actor enters valid username and password and also already registered and activated user, then he/she will be logged into the system successfully. If not the system state remains unchanged.
Extend:	N/A
Uses:	N/A
Normal Course of Events:	Actor will enter his/her username and password provided by Administrator into the Sign-in form and press Login button. System validates the username and password and redirects the actor to his/her respective home page.
Alternative Path:	If an actor isn't activated currently within the system by Administrator, then system won't let the corresponding actor to Login. Rather system will displays a message to that user "You are not activated anymore"
Exception:	If an actor enters invalid username or password, then system will displays an error.

** Use Case ID is used to make the requirements traceable*

Y. Example - 2: Filled Functional Requirements

Use Case Templates are used to generate use case diagrams

Use Case Name:	Register new patient
Use Case ID	7
Actor:	Receptionist, Administrator
Summary:	This use case describes the scenario in which actor registers a new patient within the system.
Pre-Condition:	Actor must be login prior to perform registration.
Post-Condition:	Patient will be successfully registered in the System if not already registered and will become eligible for Appointment reservation, Doctor Checkup, Lab tests and other applicable processes.
Extend:	N/A
Uses:	Patient registered, Cancel registration
Normal Course of Events:	Actor will enter all the necessary details in the patient registration form and press the “Register” button. System will save this new patient in the system and display a message “Patient Registered Successfully!”
Alternative Path:	If actor doesn’t want to register the patient, he will press the “Cancel Registration” button and system redirects the actor to its home page.
Exception:	If the intended patient already registered within the system then system will prompt a message “Patient Already Registered!” and system won’t allow registering it again.

Z. Changing Requirements

The release of first demo-able version of the software product leads to **Change Request or Bugs** from client. The changing or new requirements at this stage are managed using Change Request Template. To fulfill the new requirements, the project team always has to put some extra efforts and before incorporating any new requirement, impact analysis must be performed and the results should be communicated to the clients in order to avoid any conflict in terms of schedule, budget and/or requirement. Bugs should be fixed because they are in-scope errors which are committed to be delivered in correct form.

AA. What is Software Release?

A release can be defined as a distribution of final version of the software product. The whole software product is divided in to multiple releases which include different set of features environment variables and backlog. Features are also further divided into doable releases. In

simple words, Product development is achieved via Release when a working product is delivered to the customer.

BB. How Release is build?

Each release is considered to be a project with tangible deadlines to be followed. Features are provided by the sales or business development team while environment variables are provided by technical team. Environment Variable majorly includes next software version to be compatible with product. Backlogs are discussed and finalized between Sales and Technical Team and communicated the decision to the technical team so it can plan the release.

CC. Release Management

Release management includes the planning, scheduling and execution of the agreed features of the release. The deadlines and timelines are shared with the clients as well. The most important part is the approvals that must be seek on schedule and features to be added in the release. Release management is another name of controlling a software build throughout different stages.

DD. What are Release Notes in General?

A release note is a document distributed and released as a part of the final product/build. It contains information about the scope of the software product, performance benchmark, features added/delivered and the known issues that customer might face. The release notes are written by the technical writer but the QA team is the owner of the document. This document is only distributed once the product or service is carefully tested and approved against the requirement or specification provided by the development team.

EE. What is Software Quality Assurance?

Software quality assurance is the process to make sure that the developed product is meeting the requirements of the customer and all the features are working as per planned. SQA team is considered to be the internal customer of the product and the product is released only if approved by QA team. Software quality assurance is an ongoing process and it's a part of every stage of SDLC until the product is complete just to make sure that the required quality level is achieved throughout the development process.

FF. What is Software Requirements vs. Software Quality Assurance?

Requirements must be reviewed by the QA resources before finalizing them. QA will review all the requirements and write test cases against all the use cases. As per the modern best practices, QA must be involved since the initial stages of the project and QA must be aware of what is being built and how it's being build s so they can prepare the quality assurance activities in a planned and organized way. So it won't be wrong in saying that QA is a major stakeholder in the gathering and finalizing software requirements.

GG. Verifying Requirements

Requirement verification is a structured and organized activity in order to confirm that the built software product fully address the documented and agreed upon requirements. Quality assurance team performs various tests during all the stages of SDLC to ensure that the final product is as per the needs and requirements. To prevent rework, requirements should be validated and approved by QA and all stakeholders before development. That's why software quality assurance is considered to be an ongoing process.

HH. People Expectation from Quality Engineering

The purpose of quality engineering is to ensure that the final product being delivered is stable and, reliable and meets the requirements of client. In case of software development, people expect a product with zero bugs or defects but in reality and practicality, it is not possible to have such software product. So this kind of unreal and superficial requirement often leads to financial loss, clientage loss and alike.

Module 07: Quality Assurance Basics

Software products are not built overnight. It takes a lot of efforts, team coordination, development and testing to come up with a final version of the product. Regardless of the software product, its complexity or size, the purpose of QA remain same; to reduce the defects to minimum to ensure minimum disruption. High priority or the sensitive areas of the product should be defects free but as mentioned in the last module, it's not possible to have a software product with zero bugs. Through following standards and best practices the number of defects can be reduced to a minimum level but cannot be totally removed from the product.

A. QA and Defect

QA and defects are always tied to each other and their relationship can never vanish. QA always try to keep defects at a minimum level and for this adopts different approaches to avoid damages to the software product because of defects. The last resort is that the critical functionality of the software product should work at least and defects must not be producing any kind of threat or damage to the critical function of the product.

B. QA and Defect: Classification Scheme

Defects can be classified into three categories. These are as follows along with brief explanation of them.

- ***Defect Prevention - Error Source Removal:*** Defect Prevention is the process of addressing root causes of defects to prevent their future occurrence.
- ***Default Detection:*** it's a process to detect and remove defects as early/many as possible through various approaches like QA, code reviews, code inspection and design review.
- ***Default Containment:*** it measures the amount of defects that any QA team was able to find as a portion of total defects i.e. found by QA, appeared as run-time error. The focus of this activity is to reduce/eliminating escaping defects.

C. Pre-Release Defects

Pre-release defects are defined as **'Dormant Defects'** which have potential to create problem to users and customers. These inactive defects are triggered on a specific situation or may occur in a low priority area where the concentration of QA activities is low and they might not get specific attention until they are triggered. The following example would help in establishing a better understanding of the concept.

D. Example Pre-Release Defects

Imagine the following scenario:

'The cash register scans thousands of items such as bread, milk and cheese every day without trouble. However, when someone buys products and use cash + credit card and gift voucher to

pay for it? Since this is not common way of payment it was not tested properly and it had a bug, now whole system is in wait state due to this?'

Pre-release defects if generated at client sites then depending on impact it has to be fixed but they are usually part of release notes.

E. Post-Release Defects

Post-release defects are those who appear in production or at customer site. The software supplier company has to pay for its pocket to fix those post-release defects. Post-release defects have higher cost of fixing as compared to other defects because of multiple installation of the product and the defect has to be fixed at every place. Besides financial loss, company image, trust on the company, reputation, market position all of them will be negatively impacted.

F. Example of Post-Release Defects

Consider the following scenario:

'There was functional requirement to export logs generated from database into CSV file so user can analyze it later on. During testing CSV was tested and working fine but at Production site CSV was generated but records went missing in the files and it require hot-fix at production site.

After careful analysis it was revealed that after 500000 records in CSV more records went missing when writing it from Database and it was decided to generate separate CSV after 500000 records and in the end merge two files to generate third CSV. During QA CSV was tested for 30000 records because there was no upper_limit mention in SRS.'

Production level bugs have very high cost plus it can hit reputation also.

G. Defect Prevention Basics

Defects prevention is way to reduce the number of defects and the cost to fix. Besides it also dig deep into the sources of the error and suggest action to avoid them. Defect prevention is based on the assumption that there are known error sources or inappropriate actions causing defects and error in the software product. The best way to prevent the defect is to detect them as early as possible, find its source and take corrective actions in order to avoid them in future as well.

H. Defect Prevention: Education and Training

People factor is one of the critical factors in designing, producing and delivering quality software products. This put a lot of emphasis on team training and education because sometimes lack of skills or relevant training leads to disastrous defects that could be prevented

otherwise. Change is constant and it's fast paced and to handle changing requirements, needs, and technologies, the software engineers must be aware of the latest trends.

I. Domain Knowledge

Software developers are not considered to be the domain specialist rather they are pure technical resources. For some projects, domain specialists are hired to work with business analysts and/or developers to give some technical insights related to domain. Domain specific knowledge is of utmost important for understanding the requirements as all domains has their own terminologies and their meanings vary domain to domain. That's why low domain specific knowledge may leads to ambiguity, confusions, and demotivation among the team.

J. Example Domain Knowledge

The following scenario will help in understanding the concept:

'In Property tax calculation Project resident of the property need to provide total covered area, total constructed area, total floors, *type of property* so that accordingly *category and off-road / on-road* rates can be applied. Based on these provided data *appropriate Law* will be applied to calculate Property Tax.'

To derive appropriate rates and formula there is need of domain specialist.

K. Lack of Expertise in Phases of SDLC

To ensure delivery of Quality Software Product, adherence to best practices of SDLC is the key. Lack of execution details of Phases of SDLC is main cause of defects. Lack of knowledge in another reason behind defects as it hinders the learning and adoption of the best practices. For instance Lack of knowledge of Requirement Management or Product integration is root cause of defects. Besides there are different variation of SDLC and a person may not have sufficient knowledge of the SDLC methodology and it may affect team performance.

L. Lack of Process Knowledge

Just like lack of expertise, lack of process knowledge may give the same results. Process knowledge at team and individual level is essential to prevent defects. Any successful project required technical skills, management skills, well-defined product requirements and well-defined processes that support project performance, control and improvements. Creation of the process and on-boarding/implementation of the process are two different things. The success of process is based on how well it is defined and implemented as well. So the process knowledge is important and it absence may leads to errors and defects.

M. Example Lack of Process Knowledge

Consider the following scenario:

'During the execution of the Project and after presenting the Demo to the client if Project Manager is not fully-aware about managing the Change Request then it might end-up in asking the developer to write code which is out of scope and in this process in-scope features might get delayed or impacted, resultantly there will be defects in features which were in-scope.'

Awareness about implementation details of process is very critical for defect prevention

N. Defect Reduction

As mentioned in the previous chapters, defect prevention helps in reducing numbers of defects but it doesn't guarantee 100% defect free product. Realistically speaking there will be defects in the product no matter how much good defect prevention mechanism is adopted. But still it is very important to have a vigorous defect reduction process in-place to remove defects. The aim is to reduce defects as much possible from their sources to prevent them in future.

O. Inspection: Direct Fault Detection and Removal

Inspection is one of the most commonly used QA activities and it is also a very effective QA alternative to reduce bugs. Inspections are generally conducted to identify smells at any phase of SDLC and usually these are walkthrough of different artifacts to provide **another EYE.** Inspections are applied to code level as well to check their compliance with different artifacts.

"Software inspection deals with software defects already injected into the software system by detecting their presence through critical examination by human inspectors. As a result of this direct examination, the detected software defects are typically precisely located, and therefore can be fixed easily in the follow-up activities."[1]

P. Example of Inspection: Direct Fault Detection and Removal

The following scenario will help in understanding the concept:

'During the development phase of the Project and before QA Process to start, in Code review it was revealed there when login is performed via social media then normal login generate error. Multiple developers are working shared code-base and different developers are working on login module. After multiple inspections it was revealed that Plugin which was used for login via

[1] Tian, J. (2005). Software quality engineering: testing, quality assurance, and quantifiable improvement. John Wiley & Sons.

social media is changing status of variable at Database level which was used by team which was working normal login due to which functionality was broken of normal login. After discussions it was decided to use different plugin for login via social media.'

Decision of using plugins or off-the self-solutions should be done after careful review.

Q. Testing: Fault Detection

Test is defined as a core QA activity with the purpose to identify the faulty areas which are to be fixed. Testing is a process where QA team executes the product to find out defects or bugs and to ensure that the product gives the required output. Testing is conducted at different levels of SDLC but the purpose remains same. Formal testing of the products starts when development team hands over the code to QA team who perform different testing and sub-testing activities to ensure the stability of product.

R. Defect Containment

Despite of vigorous defect prevention and reduction strategies and activities, existence of defects is still possible and it becomes more risky in high impact applications in which the stakes of failure are very high. For example, in different software product for medical, transportation or nuclear industry, the risks associated with defects are very high. Due to modern fault detection and prevention tools and frameworks, few faults are generated under rare conditions at real-time. But still it is impossible to test exhaustively all the conditions and the final possible strategy is to break fault-failure relation to contain the damage.

Module 08: Software Quality Assurance & Defects

A. Defect Resolution

Defect Resolution is a process that keeps record of reported defect till testing department close it. The traditional methodology typically starts with defect identification by QA team during testing/QA activities, the defect is then sent to development team to fix, and then again returned to QA to be validated. But sometimes, defects are not solved due to many reasons like lack of communication, misunderstood requirements or inability to remove defects and in such cases defect are not fixed and all the parties agree to it.

B. Defect Lifecycle

A defect lifecycle consists of all the necessary steps to be carried out from defect identification or reporting to defect resolution. All the reported defects are to be fixed, re-verified and formally closed by the QA team. All the defects are tracked by assigning different statuses to them. Like as mentioned above, sometime it is not possible to remove defects, so status like Assigned, In-Progress, Fixed, NotAFix, Pending, Not Reproducible and Closed are usually used to track defect.

C. Stakeholders in Defect Lifecycle

The most important factor in defect resolution is that the decision regarding any defect must be accepted by all stakeholders. For example, if it's decided that certain defect is not to be fixed in current release, then all stakeholders must agree to that. Usually minor defects with very less impact are deferred to the next release with mutual consent and even in some cases the reported one also but can be reclassified as ‘not a bug’ if the situation demands so. The key point is that all the decision regarding defects must be made with mutual consent of all stakeholders.

D. Defect Logging

This is the very first step in defect resolution and in this step QA team tests the application or the product to identify, discover and report all the defects. A defect log is established containing all the defects identified in the testing phase with necessary parameters like defect

ID, priority and severity level, current defect status and others. Defects log also ensures that all the necessary details about defects are provided to the development team so they are able to understand and fix the defect.

E. Defect Tracking

Defect tracking is a methodology to track the discovered and reported defects and it ensures that the defects are being monitored and controlled throughout different stages of defect resolution. This is done by using templates or tools to ensure that tacking is being done properly. Today's modern software product are complex and may contain huge numbers of defects with high level of severity and criticality and this makes defect logging and tracking very important part of software development.

F. Example of Defect Logging & Tracking

Following is a sample template that can be used to log and track defects. This file is shared with all team members to improve collaboration.

Sample Defect Tracking & Log Sheet/Template

Defect No.	Date Created	Created By	Defect Description	Steps to Reproduce	PM Process	Lifecycle Phase	Priority	Owner	Assigned Date	Estimated Time to Fix	Status
1	10/19/16	Kamran	Login not working	Click on Login Button-> Valid credentials -> Error Message	Executing/ Controlling	Testing	High	Jim	10/19/16		In Progres
2	10/19/16	Saad	Logo is broken	Logo is shown distorted when loaded	Executing/ Controlling	Testing	High	Jim	10/19/16		In Progres
3											

Defect Log Instructions	
Field	**Description**
Defect Number	Unique identifier for the defect, i.e., 1,2 etc.
Date Created	Date on which the defect was initially reported and logged.
Created By	Name of the person who reported the defect.
Defect Description	Description of the defect. State the subsystem, area, or other part of the product in which the defect occurs or w
PM Process	Choose the Project Management Process this defect was reported for: Opportunity Assessment, Initiating, Planni
Lifecycle Phase	List the lifecycle this defect was reported for. The Software Development lifecycle is: Opportunity Assessment, Co Development, Testing, Documentation and Training, Deployment and Post Deployment.
Priority	The priority code indicates the impact of this defect on the project:
Owner	Name of the person who is responsible for fixing the defect.
Assigned Date	Date on which the defect was assigned for resolution.
Estimated Time To Fix	Estimated amount of time required to correct the defect. If applicable list in hours, not days.
Status	Current status of the defect: New, In Progress, Under Review and Completed.
Resolution	Description of the defect's resolution.
Resolution Date	Date on which the defect is to be resolved (or is resolved depending upon its status).
Actual Time to Fix	Actual time required to correct the defect.

G. Product Based Defects

Defects resolution in a product based scenario is an ongoing activity and remains continued even after the product is deployed on the client's site. A special team is available to provide support after deployment to ensure hot-fixes for defects removal and the support and quality maintenance plan are established in earlier stages of SDLC. Post-deployment defects are more critical in nature and cost many times more as compared to the ones rectified in earlier stages of development.

H. Project Based Defects

In a project based scenario, bugs and defects are identified, reported and fixed during the project lifecycle. In other words, all defects are to be discovered and fixed before handing over the product to customer. The defects categorized as 'NotAFix' are to be agreed by all stakeholders of the project. Service level agreements (SLA) are signed by both parties in order to provide post-deployment support.

I. Product Evaluation

As the name suggest, product evaluation is a process to evaluate the reliability, stability and the market value of the product. It can be further defines as the assessment of the final version of the software product according to the specified procedure (requirement, best practices, software development principals and alike). the key components of product evaluation includes different criteria like code quality, sustainability of the product, defect backlog, features, reliability of features, compatibility, support of the features and many others.

J. Software Maintenance and Defect Handling

As mentioned above, project and product based defect handling is different. Same is the case with support to be provided after deployment of the product. Software maintenance defines the scope of the support to be provided with all relevant terms and conditions. There are following levels of software support:

Level 1	Customer Support Team: A special team other than QA and development resources with the expertise to provide technical support to customers
Level 2	QA Team: Members from QA team are involved in providing support to customers
Level 3	Development Team: Members from development team are involved in providing support to customers

K. Tool and Processes in Defect Handling

Usually MS Excel sheets are used to log and track defects but there are many other free and paid tools available in the market for the same purpose. These tools provide that provide some

additional features to make tracking and logging easier. In those tools, support team has the admin rights and they add QA and Development resources as per the need of the situation.

L. Using Mantis for Defect Handling

Mantis is one such tool used for keeping log and tacking details of the defects. This tool is also available on cloud and it makes collaboration among team members and other stakeholders easier and faster. Some screenshots are as follows:

Dashboard View

Defects/Issues Tracking & Logging

Detailed View of Defects & Issues

M. Implementing Defect Lifecycle

Defect lifecycle consists of various customized steps – normally referred as workflows - through which a defect goes through. The defect lifecycle starts with the discovery of the defect and ends when the defect is closed and ensured that it is not reproduced. Software development firms may have different steps but tools for defect management come with pre-defined steps to manage defects lifecycle.

Module 09: Software Testing?

A. What is Software Testing?

Software testing is the process of executing the software product with the intention to ensure that it is giving the desired outcomes and satisfying all the requirements and business needs. Another purpose of the software testing is to find out defects and bugs so they are recorded, analyzed and corrected. In a layman language, software testing is all about comparing actual and the desired behavior of the product. Software testing can provide a concrete and objective view of the quality of the product.

B. Rationale for Testing

In a bigger picture, the purpose of testing is to ensure that system is working as expected and it is one of the most important parts of software quality assurance activities. Testing provides objective information about the quality and allows stakeholder to put confidence in the software product. It also reveals the risks, if any, associate with the implementation of the product. The most natural way is to dry-run the software in controlled or simulated environment so the software defects/bugs are taken care of before deployment of the product.

C. Artifact Level Testing

The primary software artifact to be tested is the software program or code written in different programming language. Testing is usually performed by writing and executing test cases which are derived from use cases. Test case is document that describes the testing activities, pre and post condition, expected system behavior and result for a specific test. While use cases describe all the steps that final user will be performing to complete the task and those steps become input of the test cases which elaborate those steps and actual & expected results against each one.

D. Major Activities in Software Testing

There are three main activities in software testing and these are described below with brief explanation:

No.	Activity	Explanation
1	Test Planning & Preparation	The objective is to set the goals for testing, select the best and appropriate testing strategy/methodology, and prepare specific test cases and the general test procedure.
2	Test Execution	Execution and implementation of the testing strategy and measurement of product behavior.
3	Analysis & Follow-up	This step includes analysis of results to determine if a failure has been observed, and if so, corrective actions are taken and monitored to ensure removal and reproduction of defects.

These activities are summed up in the following image. This is to keep in mind that defect fixing and re-verification both are part of software testing.

E. Functional Testing

Functional testing is a type of software testing which is more concerned about verifying that the software and all of its function are performing in total conformance with the software requirements. Its focus is more on the external behavior of the software i.e. to test the functionality and the features of the product how functionality is achieved i.e. programming or coding is not a matter of question in functional testing. It is usually called as abstract level testing because it just ensures that software is giving desired output to the end user.

F. Structural Testing

This is the total opposite of the functional testing and it is more focused on the internal implementation details of the software product. The purpose of this type is to test the different programming structures and data structures used in the program. Structural testing verifies that the internal units of the software product like programming/code structure, blocks, data structure and all relevant interconnections are implemented correctly. This is concrete testing from code view point because Code is tested at statement level or module level or sub-system level.

G. Black Box Testing

Functional testing and black box testing are two terms used interchangeably. Functional testing is usually achieved by performing black box testing. In the most simplest for, black box testing is an ad-hoc running and execution of the software system to observe any difference between expected and actual performance and in case of gap or variance from the desired results this process is repeated to eliminate the possibility of hardware failure causing the difference.

Specification checklist is a good way to perform this testing. Specification checklist describes the functionality and features of the product with sample input and expected output data. Smoke Tests are most abstract level of testing that aims at ensuring that the most important functions work.

H. Example 1: Black Box Testing

Consider the following feature of a software product to perform arithmetic operation on numbers:

This software takes some input and gives some output and what's going on inside is unknown. Some of the test ideas would be something like:

- To explore supported values, perform test for floating point number, integers, and boundaries of integers.
- In case of division, Divide by 0 would be a classic error.
- Use mouse, keyboard, type and paste to explore the possibility of entering values.

I. Example 2: Black Box Testing

Imagine we are testing a Date Class with a DaysInMonth (month, Year) method. What are some condition and boundary test for this method?

Possible answers include:

- Check for leap year (every 4th year)
- Try years such as: even 100s, 101s, 4s, 5s
- Try months such as: June, July, Feb, Invalid Values

J. White Box Testing

Structural testing is achieved by performing white box testing and its aim is to verify the correct internal implementations i.e. programming/code, data structures, blocks, relationships and interconnections are implemented properly. This test is done by executing and observing the program/software behavior related to these specific units. Different debugging tools are used to test coverage of specific statement level code. This means that tester is able to see if all or a specific statement has been executed or not, and if yes, is it working properly and giving the desired output or not. In this way the error is identified with its location. White box testing is performed by developer himself as it required extensive knowledge of code and flow of code. White box testing covers branch level testing (if-else, loops, switch) and Path level testing (all possible paths).

K. Example 1: White Box Testing

Statement Level and Branch Level testing ensure coverage to statements and branches (if-else, while, for, switch).

```
Float calculate (int a, int b, int c)
 {
   Float e=0.0
   if (a==0)
   return e;
   int x=0;
   If (a>b && c<b&&c>a) –- a=4, b=3, c=2
{
 X=a;}
 E=1/x;
 return e; }
```

L. Example 2: White Box Testing

Path Coverage: All the possible paths (decision points to be tested). Path Level Testing ensures maximum coverage of code.

```
Int main ()
{
  Int a,b,c;
  C=a+c;
  If (c>100)
  { Cout <<"It's done" ;}
  If (a>50)
  {Cout<<"It's Pending" ;}
}
```

TestCase_01: A=50, B=60
TestCase_02: A=55, B=40
TestCase_03: A=40, B=65
TestCase_04: A=30, B=30

M. Reviews in Testing

Reviews is another kind of software testing which is performed before executing the test cases or executing the software code to see if its working properly. This is a static testing method in which the software artifact like code is manually examined without execution with the goal of early defect detection and removal. In this way, the mistakes overlooked in the early stages of software development are identified and corrected which ultimately improve the software quality.

N. Rationale for Review

Reviews in context of testing are time saving activity and need no formal planning or resource utilization because it is more of an informal testing. Defects identified in this stage are easy and less costly to handle and rectified as compared to defects identified in formal types of testing. Reviews include code and design walkthrough and it is usually done by more experience technical person who is not the author of code. It can take shape of informal code walkthrough that increase the knowledge sharing as well.

O. Inspection

It is more formal type of static testing (Reviews). In inspection, the review activity is normally led by a trained facilitator or moderator who is not the author of the code. This formal activity aims at evaluating the level of compliance with specific rules, principles or the best practices of software development. All inspections activities are well documented and all the defects identified in this stage are logged in defect logging sheet.

P. Example of Inspection

Inspection for Code is basically removal of redundant or unwanted code at max. Following are some of the question asked during inspection activity.

- Is the compilation listing free of warning messages?
- Are there any uncalled or unneeded procedures?
- Can any code be replaced by calls to external reusable components or library functions?
- Will requirements on execution time be met?
- Is the code well-structured, consistent in style, and consistently indented?
- Are there any blocks of repeated code that could be condensed into a single procedure?
- Are symbolic used rather than "magic number" constants or string constants?

Q. Test Check List

Test Checklists are steps to be executed every time testing is to be performed. It consists of standard steps to list down all activities and tasks to be performed before the test and during the test. This is not a kind of functional testing which is totally different. This is simple a list of activities to be performed to ensure the readiness for formal QA tests.

R. Example 1: Test Check List

Below is a sample **web testing checklist for data security**. These are only sample questions and may include other as well.

- Are data inputs adequately filtered?
- Are data access privileges identified? (e.g., read, write, update and query)
- Are data access privileges enforced?
- Have data backup and restore processes been defined?
- Have data backup and restore processes been tested?
- Have sensitive and critical data been allocated to secure locations?
- Have date archival and retrieval procedures been defined?

S. Example 2: Test Check List

Below is a sample web testing checklist for performance. This is to be noted that check list does not include functional testing. Testing Checklist is very critical to test overall behavior of the System.

- Has the database capacity been identified?
- Has anticipated growth data been obtained?
- Is the database self-contained?
- Is the system architecture defined?
    - Tiers
    - Servers
    - Network
- Have the various environments been created?
- Is load balancing available?

Module 10: Test Activities and Management

A. What is Configuration Management?

In software engineering, configuration management is the collection of the best practices of tacking and controlling changes in the software development. Multiple teams and resources work on different parts of the software at same time and their work need constant changes. So configuration management team helps them in keeping record of different versions of their work and determining what was changes and who changed it and make them able to revisit the changes if needed. Configuration management is closely associated with QA as QA team receives shipment of software code from configuration management team and vice versa.

B. Why Configuration Management

Multiple developers and programmers work on different part of the software on a shared code base towards single working software, different stages like development, testing and production required different version of the same software and SCM make sure that a specific version is available with all of its changes history. CM is intended to eliminate the confusion and error brought about by the existence of different versions of artifacts. There are many software products available in the market like VSS, TFS and Git that are used to manage changes and different version of the code.

C. Example of Configuration Management

The need is to have a single repository to manage all the releases of the software and CM is tasked with fulfilling of this need. All the release activities are coordinated by SCM team and then the codes base is handed over to QA and then to customer is passed. But customer may not like changes made in a certain release, as in release 2 in the image on right. Thus, versioning and keeping backup of all the releases makes it easy to revert back any changes. So SCM also manages what changes are requested by which customer.

D. Product and Configuration Management

A single software product may have different version based on the clients need and all the versions may have multiple codes bases. Changes requests made by one client is not visible to other clients and those changes must not impact other clients as well. So it is the task of SCM to manage customer requests and determine what version of software is being used by that client. Without Configuration Management it is impossible to track multiple software versions and their multiple code bases. Changes in software products are facts either for error removal or just for product refinement and it is the responsibility of SCM to manage all changes in complete details.

E. Test Planning and Preparation

This is one of the most important activities in the whole testing process because most the decision related to testing are made in this stage which determine the quality of the software testing. The decisions made in this stage are based on several key questions and answers to those questions decide the testing strategy. Few of those questions are as follows:

- What are the objectives and overall goals of the testing activities?
- What specific areas/features of software are to be tested?
- What would be the exit criteria or when to stop test?
- In what environment(s) the software is to be tested?
- There are many other questions as well that set the direction of software testing

F. Test Cases Basics

As mentioned above, test case is document that describes the testing activities, pre and post condition, expected system behavior and result for a specific test. All the software requirements are further divided into use cases which describe what users will follow to perform a certain task. Test cases are then established against all user stories to verify that whether users will be able to perform them or not. Test cases guide the tester through a sequence of steps to validate whether a software application is free of bugs and working as required by the end user.

G. Writing Test Cases

A well written test case allow user to understand and execute the test. There are few best practices that must be followed when writing test cases and some of the mare as follows:

- Each test case should test only one test condition at a time.
- Multiple test cases should not be tested in single test case.
- Ensure coverage of positive and negative scenarios and include all the necessary details.

- Test cases must include detailed test steps all necessary data and information on how to execute the test.
- Expected and actual results must be added in the test cases and in case of variation from the desired results, defect must be logged which go through the defect lifecycle and the tester will again verify it once fixed.
- Should be repeatable
- Complex language should be avoided in order to remove and ambiguities and difficulties in understanding the test cases.
- Test cases should be traceable to requirements (Use Case)

H. Test Case Template

Following template can be used to write test cases. In the following template, comments section can be used to refer to Use Case # if required.

Sample Test Case Template

Project Name	
Module Name (In case of Product)	

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail	Comments

I. Test Cases vs. Functional Requirements

Every software product has some functional requirements stating the nature of function it will perform or what the system is supposed to accomplish. Test case defined all the steps to test the functionality of the product. In case of closure of the test, the product is considered to be providing all functionalities as per the requirement. Test Cases can refer to multiple functionality depending on use Cases. Following examples will help in completely understating the concept of test cases.

J. Example 1: Test Case Template

Project Name	XYZ
Module Name (In case of Product)	NA

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail	Comments
TU01	Check Customer Login with valid Data	i. Go to site http://www.test.com ii. Enter User Id iii. Enter Password iv. Click Submit	User id = guru99 Password = pass99	User should Login into application	As Expected	Pass	
Note: There can be multiple tests against one scenario.							

K. Example 2: Test Case Template

Project Name	XYZ
Module Name (In case of Product)	NA

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail	Comments

TU02	Check Forget Password	i. Go to www.test.com ii. Click on Login Page iii. Click on Forget Password Link iv. Email Text box to PopUp v. Enter Valid Email vi. Reset Password URL to be email vii. User Click on Reset Password URL viii. User Should set password and conform new Password ix. Customer should be able to login with new Password	Email for reset Password sherazpervaiz @gmail.com	User should be able to login with new password	Failed	Failed	Step - ix Fail
Note: If individual Step Fail then overall test case Fails.							

L. Example 3: Test Case Template

Project Name	XYZ
Module Name (In case of Product)	NA

Test Case ID	**Test Scenario**	**Test Steps**	**Test Data**	**Expected Results**	**Actual Results**	**Pass/Fail**	**Comments**
TU03	Verifying Dashboard for Product	i. Go to www.test.com ii. Register iii. Login iv. Ad new Product	UserName: Admin Pwd: Admin	Product Dashboard Count should be updated with addition of new Product	Product Dashboard is updated	Pass	
Note: There can be multiple tests against one scenario.							

M. Product Testing vs. Project Testing

The project and product based testing have different features but the core testing activities are generally same for both. Here are some differences in project and product based testing.

Project Based Testing	**Product Based Testing**
A project is a software application that is developed by a company with the budget & requirements from a	*A product is a software application that is developed by a company with its own budget & requirements from*

customer.	*market standards.*
Project Testing is for client and based on requirement from clients.	Product Testing is for organization own Product and based on organization's own requirements and market standards.
Primarily project testing is a one-time activity and ends when product is delivered to client.	Product Testing is on-going activity.
Test cases in Projects are usually specific under specific environment	Test cases in Product are regularly updated as per release

N. Unit Testing

Unit is a smallest and testable part of software like functions, classes, or interfaces. Usually it has one or more inputs and single output. Unit testing is a level of software testing where individual units of software are tested and algorithms are verified independently. The purpose is to validate that each unit of the software is performing as per requirements and design. This is considered to be the very first level of testing and more of a development process as it is usually performed by developers itself. As units are tested individually and independent, unit test provide a cheap way to verify the functionality without impacting other functionality.

O. Example 1: Unit Testing

Unit Tests ensure stability and functionality of software from beginning, even before the start of formal testing.

```
public class Calculator {
        public int add (int x, int y)
        {
                return x+y;
        }
        public int sub (int x, int y)
        {
                return x-y;
        }}
public class test
{
Public static void main(String [] args)
{
Int result=0;
Calculator A = new A();

Result=a.add(5,9);
System.out.println(result);

Result=a.sub(5,9);
System.out.println(result);
```

```
}}
```

P. Impact of Unit Testing

Unit testing ensures early defect removal as it is very easy to look for defects individually and independently in all units of software code. As it is done in isolation, it becomes easy to debug the code to prevent defect reproduction and the unit test efforts are done at developer's level who can easily understand the code. Further, the cost of fixing a defect detected during unit testing is lesser in comparison to that of defects detected at higher levels.

Q. Sanity (Integration) Testing

The next level of testing is sanity in which multiple units are tested together. As mention in the configuration management section, multiple developers work on different modules of the same software and then SCM merge the code. That merged code is then handed over to developers again to perform sanity testing. The bottom line is, if the units are working individually and independently, they should be working in combination too and should be performing as expected. The focus of sanity testing is on integration of different components to work together so components with their interconnections are examined and evaluated.

R. Example 1: Sanity (Integration) Testing

Let's imagine a product database in which a user can login to add products and products categories and can search the previous added products as well.

Unit 1: Before Integration

Unit 2: Before Integration

These two units are working independently and they must work when integrated with each other.

*** Search Product to work if login via social medua

S. Sanity Testing Reporting

Sanity test reporting provides overall status of the integration testing and it helps in identifying critical areas that are to be fixed. Features are identified with data that require bug fixing. The bugs are identified as critical, major, medium and cosmetic for all the units/features. The following example will help in understand how data is recorded for sanity tests.

T. Example 1: Sanity Testing Reporting

As the following image explain, let's imagine a product with 4 features namely Registration, Booking, Payment & Reports. Sanity or integration testing performed on the mentioned features might look like the image below in which bugs/defects are identified as per their severity level.

Reflect integrated status of multiple features

U. Feature Level Testing

Functions of any software product are functional requirements that are developed as a unit and depending on feature size and nature there can be multiple underlying units. Feature level testing is aimed at making sure that software and its features works properly to meet all the intended specifications and requirements. In simple words, it's a test for software functionality where each function, the might be consisted on multiple units, is evaluated to see if it's is working as required and designed.

V. Example 1: Feature Level Testing

Imagine a product with a feature that enables a user to book a hotel room via website. That feature will be consisted of the following units.

- **Feature Name: Booking of Hotel Room via Website**
    - Registration of User
    - Searching of Rooms in Hotel
    - Selecting a particular Room in a Hotel
    - Filling – in payment details
    - Verification of Payment Details
    - Generation of Unique Booking Id
    - Sending Booking ID to User via Email

Each feature can be developed as one or multiple units and in feature level testing all features of the product is to be tested. Feature level testing require integration testing to test feature if there are multiple units.

W. Example 2: Feature Level Testing

Consider another product with hierarchy management feature. The units for this feature will be as follows and they all will be test in feature level testing.

- **Feature Name: Hierarchy Management**
  - Register Hierarchy
  - Add Hierarchy Name
  - Add Hierarchy Level
  - Add Root of Hierarchy
  - Root Hierarchy can have child
  - Child can be root node also

X. System Level Testing

This is and advance level of testing and the fully integrated final product is tested and evaluated in exact environment in which it is to be used by the end users. The overall architecture of the product is evaluated along with its ability to satisfy the business requirements. Software evaluation is done on the basis of user's point of view and has very less to do with code design and structure. System testing is performed by independent testers who do not have any contribution in development activities. System Testing is very important because it verifies that the application meets the technical, functional, and business requirements that were set by the customer.

Y. Stress Testing

It is another form of system level testing and the purpose of this test is to evaluate the effectiveness and stability of the software under unfavorable conditions and that mostly include quantitative test to gather stats related to frequency of crash and errors. During the test, adverse and unfavorable conditions are deliberately created and maintained and fully integrated system is evaluated throughout those environments to measure their breaking point. A fine example is clicking a website with multiple concurrent to test the load.

Z. Regression Testing

This test is performed to verify that with the addition of new feature the previous developed and tested system still perform correctly. The complete software development project includes maintenance, defect fixing, optimizations, error correction and that result in continuous update in the product. The goal is to make sure that with the addition or change in one part does not impact other parts and the newly added features are fully integrated with the already developed system. The regression testing verifies that the goal is achieved or not.

AA. Example 1: Regression Testing

Lest consider development of a product and the following features were developed in the release 1.0 and 1.1.

Feature No.	Release 1.0	Release 1.1
1	Login	Login
2	Product Listing	Product Listing
3	Shopping Cart	Shopping Cart
4	Payment via Credit Card Only	Payment with Discount Card using PayPal also
5	---	Discount Card

In release 1.1 two features were added or updated (discount card and payment option with discount card or PayPal) and that should not impact the already developed features.

BB. Test Reporting

Test reporting is the most important artifact of the whole software testing activities as it shows the overall status of all the testing activities performed throughout all stage. There are different levels of tests at different stages and it becomes extremely important to track all tests performed and their results. Many different shapes and formats are available for overall status of test activities and one such format is Test Dashboard that is described in following lines.

CC. Test Dashboard

Test dashboards are statistical analysis of overall testing activity in progress. The data is statistically and visually presented in this dashboard and they help testers a lot in making useful decision regarding testing strategy. By using the Test dashboard, the QA tea, can monitor test activities, report on progress, find gaps in test coverage, and identify test areas that may require additional investigation.

DD. Example 1: Test Dashboard

The test dashboards are able to tell the count of defects of all statuses (closed, fixed, hold, NotAFix etc.) separately for all the features of the product, as shown in the image below.

EE. Example 2: Test Dashboard

The data in the test dashboards can be graphically presented to make it easy to understand. The example is shown below.

FF. Testing: Release Notes Basics

Release notes are documents that are distributed with the software release revealing what is included in the software i.e. the features and steps on how to install and deploy the product. The list of known defects that are not fixed is a part of release notes while mentioning what defects, deficiencies are fixed in a particular release. Another important thing that this report contains is test results and information about the test procedure to increase client or end user confidence in the product.

GG. Example 1: Release Notes

HH. Example 2: Release Notes

II. Post Production Issues

Post Production issues are reported be end-users once Staging Environment is Productionize, productionalizing an environment mean to deploy staging server codebase to Production Server which is provided be client or client has some sort of arrangement with Vendor for deployment. Usually good QA will ensure that there will no critical bug on Production side but even good QA will not ensure 100% bug free code that's just possible, production issues are those issues which are reported be end-users can be of low priority from development but from end-user viewpoint they might take high-priority which are needed to be addressed.

JJ. Types of Post - Production Issues

The way to handle pre-production issues and post-production issues is different simply due to the reason that end-user is usually not involved during development. During development in-scope bugs are fixed and shipped to ensure at max compliance with

agreed requirement. In Production it is possible that End-user or client may report issue which is not a bug but needed to be fixed i-e WishList or issue is there due to latest build - Ripple Effect. For WishList vendors usually do analysis, estimates and share timelines but these are paid changes but for Ripple Effect cost is bear by Vendor because somehow bug gone undetected during QA. First step to proceed when issue is reported from Production is to classify it either a Bug or Wish List and then accordingly steps are taken.

KK. Hot Fixes:

Hot fixes are issues reported from Production (Live Site) which can either be WishList or Bug but in either case we have little time to respond for Hot Fix due to importance and criticality. There are scenarios where end-user or client is not worried about cost factor and just expect to fix issue asap as it may be impacting business, in this scenario usually complete QA process is not executed majorly due to time factor and team is dependent on unit testing and Code Reviews to fix hot fixes rather than regression testing.

LL. Patch:

Patch as opposite to Hot Fix are those issues which are critical but client can wait for short period of time to get them fixed. Difference between Patch and Release is that for timeframe for Patch to deliver is usually less than Release but more than Patch and dev team do proper estimation which include QA cycles also and then share deadline. Usually leftover bugs or WishLists or both are delivered as patch after Major Release.

MM. Example of Patch

Assuming there was requirement from client to develop a website where user registration will be performed by clicking on registration page and then user will fill-out registration form and submit the details and then user will be able to login with credentials. After Project is delivered to the client , there is need to integrate registration using Social Media, catch is to identify this request either as WishList or bug, in this situation Specs document or SRS because to negotiate with client verbal communication are usually enough to convince client in case of conflict. In this scenario is client agree it as a WishList then estimates and deadlines will be shared accordingly.

Appendix – I:

i. Technical Design Document