**Notes for MCA-II (Semester- III)**

**Subject :- Software Testing & Quality Assurance**

**(Subject Code:- IT-33)**

**Chapter: 1] Software Quality Assurance Fundamentals**

**1.1 Definition of Quality, QA, QC, SQA**

Quality cannot be assured without first understanding its nature and characteristics. So the first question one has to ask is: **what is quality?**

Quality from

1. A customer view point fit for use or other customer needs.
2. Producer’s viewpoint meeting requirements.

Software quality is defined as *conformance to explicitly stated functional and nonfunctional requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.*

This definition emphasizes upon three important points:

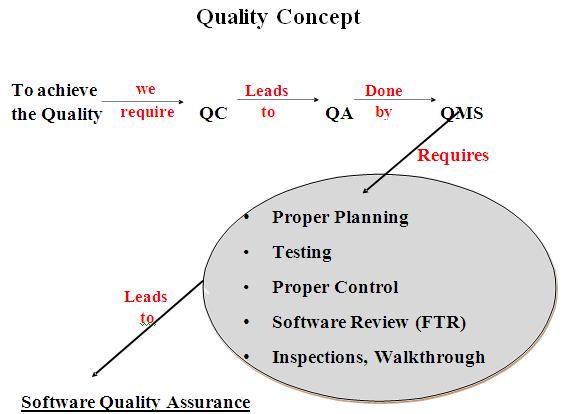
* Software requirements are the foundation from which quality is measured. Lack of conformance is lack of quality
* Specified standards define a set of development criteria that guide the manner in which software is engineered. If the criteria are not followed, lack of quality will almost surely result.
* A set of implicit requirements often goes unmentioned (ease of use, good maintainability etc.)

The totality of features and characteristics of a product or service that bear on its ability to satisfy given needs.

A predictable degree of uniformity & dependability to low cost and suited to the market. (By Dr. Edward Deming)

Quality is conformance to requirements. (Philips Crosby)

Quality is a degree to which a set of inherent characteristics fulfills the requirements. [ISO 9000]



**Figure 1.1: Quality Concept**

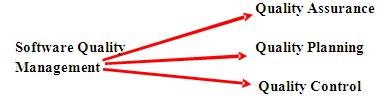
**Quality Assurance:**

*Definition:*

A planned and systematic pattern of all actions which is necessary to provide adequate confidence that material, data, supplies & services conform to established technical requirements and achieve satisfactory performance.

Quality assurance consist of a set of auditing and reporting functions that access the effectiveness of quality control activities.

Goal of quality assurance is to provide the management with the necessary data to be informed about product quality. It consists of auditing and reporting functions of management. If data provided through QA identifies problems, the management deploys the necessary resources to fix it and hence achieves desired quality control.



**Figure : Software Quality Management**

**Quality Assurance:**

The establishment of a framework of organizational procedures and standards which leads to high quality software.

**Quality Planning:**

Selection of appropriate procedures and standards from the framework and the adoption of these procedures/standards for a specific software project.

**Quality Control:**

The definition & enactment of processes which ensures that the project quality procedures & standards are followed by the software development team.

QC activities are work product oriented. They measures the product The direct result of these activities are changes the product.

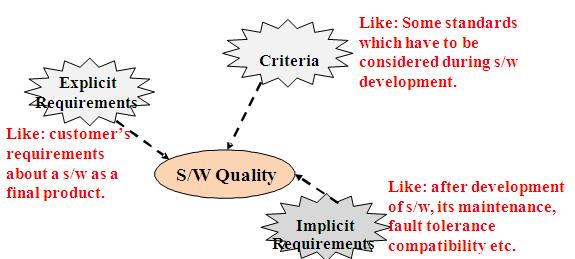
The next question that we need to ask is, once we have defined how to assess quality, how are we going to make sure that our processes deliver the product with the desired quality. That is, how are we going to control the quality of the product?

The basic principle of quality control is to control the variation as variation control is the heart of quality control. It includes resource and time estimation, test coverage, variation in number of bugs, and variation in support.

From one project to another we want to minimize the predicted resources needed to complete a project and calendar time. This involves a series of inspection, reviews, and tests and includes feedback loop. So quality control is a combination of measurement and feedback and combination of automated tools and manual interaction.

**Software Quality Assurance:** (According to Pressman)

Conformance to explicitly stated functional & performance requirements, explicitly documented development standards & implicit requirements, & is expected from all professionally developed software.

****

**Figure : Software Quality Assurance**

* **Difference between Quality Assurance (QA) and Quality Control (QC)**

|  |  |
| --- | --- |
| **Quality Assurance (QA)** | **Quality Control (QC)** |
| * It is a procedure that focuses on providing assurance that quality requested will be achieved | * It is a procedure that focuses on fulfilling the quality requested. |
| * QA aims to prevent the defect | * QC aims to identify and fix defects |
| * It is a method to manage the quality- Verification | * It is a method to verify the quality-Validation |
| * It does not involve executing the program | * It always involves executing a program |
| * It’s a Preventive technique | * It’s a Corrective technique |
| * It’s a Proactive measure | * It’s a Reactive measure |
| * It is the procedure to create the deliverables | * It is the procedure to verify that deliverables |
| * QA involves in full software development life cycle | * QC involves in full software testing life cycle |
| * In order to meet the customer requirements, QA defines standards and methodologies | * QC confirms that the standards are followed while working on the product |
| * It is performed before Quality Control | * It is performed only after QA activity is done |
| * It is a Low-Level Activity, it can identify an error and mistakes which QC cannot | * It is a High-Level Activity, it can identify an error that QA cannot |
| * Its main motive is to prevent defects in the system. It is a less time-consuming activity | * Its main motive is to identify defects or bugs in the system. It is a more time-consuming activity |
| * QA ensures that everything is executed in the right way, and that is why it falls under verification activity | * QC ensures that whatever we have done is as per the requirement, and that is why it falls under validation activity |
| * It requires the involvement of the whole team | * It requires the involvement of the Testing team |
| * The statistical technique applied on QA is known as SPC or Statistical Process Control (SPC) | * The statistical technique applied to QC is known as SQC or Statistical Quality Control |

* **KEY DIFFERENCE**
* Quality Assurance is aimed to avoid the defect whereas Quality control is aimed to identify and fix the defects.
* Quality Assurance provides assurance that quality requested will be achieved whereas Quality Control is a procedure that focuses on fulfilling the quality requested.
* Quality Assurance is done in software development life cycle whereas Quality Control is done in software testing life cycle.
* Quality Assurance is a proactive measure whereas Quality Control is a Reactive measure.
* Quality Assurance requires the involvement of all team members whereas Quality Control needs only testing team.
* Quality Assurance is performed before Quality Control.
* Software Quality Assurance Challenges

**1. Facilitation of Quality**

To accept a system for testing software, it’s extremely important to understand its nature. This is why product owners must join hands QA and testing professionals so they can test the product according to the business model. QA teams need to give their input in the initial discussions because that’s necessary in facilitating quality.

**2. QA Culture**

The QA culture plays a key role in streamlining QA processes. Companies need to change their testing culture and explore innovative ideas. Without experimenting with unique techniques and different QA testing tools, it’s difficult for organizations to test software products efficiently and quickly. Developing a comprehensive QA is necessary to meet market demands and deliver quality.

**3. Early Testing**

Similar to development engineers, QA and testing professionals have a crucial role in the Software Development Lifecycle. If you begin testing in the early stages of the development cycle, you can identify several quality issues throughout the whole cycle more effectively.

It’s easier to give valuable early feedbacks and offer solutions on features, the scope of the project, sustainable development, and architecture. Each one of these elements cuts high additional expenses that are a result of late detection of errors. At the same time, it reduces unnecessary delays in time-to-market, giving teams a chance to compete more effectively.

However, as your software quality assurance needs to test code frequently, it’s hard to document everything accurately. You need a fully committed and dedicated testing team that can create self-sufficient stories for testing. Although implementing testing early is not easy, the benefits of early testing outweigh the several challenges it poses.

**4. The Merge of Development with Testing**

To build an effective development team, organizations need to mix the development teams with software quality assurance teams in terms of knowledge sharing. That being said, QA and testing professionals should be encouraged to have some knowledge about the development process of a product while the developers should be encouraged to take active parts in enhancing test automation infrastructure and should possess knowledge about different QA testing tools.

Knowledge sharing is the key in this situation. Organizations must therefore make the development team work together with the software assurance team. This will result in a developed and effective quality development team.

Developers must learn about the testing process. On the other hand, testers must also have sufficient knowledge about software development. Hence, this will ensure us that the software we are using is definitely of good quality.

**5. Time**

When it comes to the time it takes to test, device and browser coverage, developers taking up QA efforts, and the small number of hours in a day all add up. If a team needs to test for all of the newest web and mobile devices and browsers, the team is either continually stuck with the cost of updating their device farm, or they’re relying on simulation software to provide accurate representations of the customer experience. Not only is the former expensive, it’s also time consuming. On the other hand, the latter is more scalable but also a less accurate depiction of customer experience.

Additionally, QA can seriously become a bottleneck for engineering teams when developers are stuck combing through a product for bugs as opposed to focusing on the next build. Developers should being looking ahead at what’s next, not stuck critiquing their own creations. Lastly, even with a fully scaled QA process or in-house team, there are only so many hours in the day and only so many hours that can occupy the time of dedicated QA resources. Maybe you need testing overnight, over the weekend, or over the holidays? These duties are simply difficult to impart upon an in-house team.

**6. People**

Moreover, one could argue for a connection between the comprehensiveness of testing and the amount of personnel available to provide this testing. Scalability is a critical factor in software testing, as the number of testers per test cycle can impact both the time it takes to receive results as well as the quality and overall completeness of findings. Even if a team does have a robust personnel count dedicated to QA, how often are tests running? Eventually, any team will hit capacity, and there will be a backup of scheduled tests. One way of providing against this is with an unlimited testing model

**7. Automation**

Many teams don’t have automation in place yet and assume it to be the answer to everything. They may also lack the time or resources to get it running efficiently. Many teams don’t have the time to manage automation and subsequent documentation when they’re trying to move quickly. Every time a minor change is made, a given test case might be affected. This often necessitates that someone be hands-on-deck for all subsequent changes. Even issues as simple as broken links (sometimes due to changes in content or functionality on the website), can be hard to keep up with without a large in-house QA team keeping up-to-date on all, even minor, product changes.

If conversations with our prospective customers have taught us anything, it’s that if we want to see real improvement in the QA process, we need to affect the common challenges facing most teams. We need decrease the time required to test effectively, minimize the amount of internal employees needed to test completely, and properly support the development of automation processes. No QA process is perfect, but changes in QA technology

Technology is developing fast. The field of software quality assurance has grown rapidly. This growth is making developers face a lot of challenges.

Despite of these challenges, developers were not fazed. On the contrary, they made developers more determined to do good with their job. What challenges did SQA face?

**8. Test Coverage**

Requirements change as fast as technology develops. There is always the possibility that important testing functions will be missed.

Because of this, test requirements must be traced thoroughly. Maintaining traceability of requirements is most important.

**9. Quality Assurance Culture**

Quality assurance companies need to explore. In fact, the testing and quality assurance culture must experience change. This change will be resulting in performing SQA activities properly.

Finding unique techniques is a way of dealing with this challenge. Exploring on innovative ideas will help. Product testing will be then be done faster and efficiently. Being able to meet the demand for high quality products in the market will be ensured.

**10. Build Verification**

Frequent builds may increase the possibility of code breaking existing features. Automated testing should therefore be used then. Therefore it is not advisable to use manual testing.

**11. Facilitation Of Quality**

It is very important for a quality assurance team to understand that a system needs to be verified. The nature of the business must be considered hence. The Quality assurance team must coordinate and work together with product owners and business experts.

The quality assurance team must be involved in discussing about important concerns. This will consequently make them facilitate product quality.

**12. Collaboration**

Another challenge that the quality assurance team may encounter is collaboration. Involving the team from the early stages of development which, as a result, lead to proper collaboration. Hence making software developed and supported effectively.

**13. Skill Gaps**

Sometimes, your team members have gaps in their skills and the type of expertise the project demands. As a QA project manager, you have to identify what skills your testing resources lack, so you can create an appropriate training plan for them. For instance, if some of your team members have to improve their testing skills, you should help them close that skill gap.

**14. Training and Assessment**

In human resource planning, you should consider how existing members are trained and developed to gain the required skills and develop expertise. QA project managers need to create a training plan and apply it promptly to fill in the talent gap.

**15. Evaluation**

Putting your testing resources is not sufficient on its own. Instead, it’s important for QA project managers to monitor and evaluate these programs frequently, so they can ensure they are effective.

For instance, you can work on your developer’s training expertise by assessing his progress and assigning him different tasks to see if he has learned from the training initiatives. If the developer is finds testing difficult, the manager should either replace him or consider an alternative training method.

Other challenges :-

* 1. Communication at all levels
  2. **Unrealistic Schedule**
  3. **Unclear Requirements / Changing Requirements**
  4. **Unclear Expectations / Outcome**
  5. **Utilization of Automation Tools**
  6. **Delivering a product with good quality on time**
  7. **Managing Metrics to provide visibility to customer**
  8. **Focus more on Customer Business, domain and market**
  9. **Lack of Resources / Expertise’s**
  10. **Internal Issues within the Team**

**1.2 SQA Planning & standards:**

* SQA plan provides a road map for SQA.
* SQA Plan should include:

1. System Design Review

2. S/W requirements specification review

3. Preliminary design review

4. Detail design review (Module Level review)

5. Review of integration test plan

6. Code Review

7. Review of test procedures

8. Audit of document standards

9. Configuration Control Audit

10. Test Audit

11. Defect data collection, evaluation analysis

12. Tool certification

13. Vendor & contractor oversight

14. Record keeping

* **SQA standard recommends the structure for planning to identify:**

1. The purpose & the scope of the plan

2. A description of all s/w engineering work products (documents, code etc)

3. All applicable standards for s/w process

4. SQA actions & tasks

5. Tools & methods supporting to SQA activities

6. S/w configuration mgmt procedures for managing change

7. Methods for assembling, safeguarding & maintaining all SQA related records

8. Organization roles & responsibilities relative to product quality.

* **Some SQA related standards as published by US DOD (dept of defense) are:**

1. IEEE Std. 730 : IEEE standards for establishing SQA plans

2. DOD-STD-2168 : A s/w quality evaluation standard

3. DOD-STD-2167 : Standard for all aspects of s/w engineering

4. DLAM-8200.1 : Defense Logistics Agency Manual gives guidelines for

government audit for SQA activities

5. MIL-S-52779A : The basic SQA specifications for procurements.

**Quality Standards:**

* Standards involved in QA: Product & Process Standards
* Standards involved in QA: ISO 9000 Q standard
* Standards involved in QA: CMM
* Standards involved in QA: Six Sigma
* **Standards involved in QA: *Product & Process Standards:***

|  |  |
| --- | --- |
| **Product Standard** | **Process Standard** |
| 1. These are the standard that applies to S/W product being developed. | 1. They include specific process activity which assures that the product standards are followed. |
| 2. Include documentation standards (related to structure) and coding standards  (Programming features). | 2. Used during the s/w development and includes: definition of specifications, design & validation process + description of documents which is generated during process. |
| 3. Applied to s/w process o/p. | 3. Applied on process itself. |
| 4. Product standards are related to Design Review Form, Program style, Requirement document & project plan format structure. | 4. Design Review conduct, Project plan approval process, change control process , Test Process. |
| 5. Concerned with product quality factors : cost, time & schedule, people quality, development technology & process quality. | 5. Concerned with process quality factors : understandability, visibility, supportability, acceptability, rapidity, reliability, robustness, maintainability |

* **Standards involved in QA : *ISO 9000 Q standard***
* According to this standard, a quality assurance system may be defined as the organizational structure, responsibilities, procedures & resources for implementing quality management.
* Quality assurance systems are created to help organizations to ensure that their products & services satisfy the customer expectations.
* ISO 9000 describes the general guidelines for QMS & is applicable to any business.
* ISO 9001:2000 QS standards apply to SE.
* Areas covered by ISO 9001 model for quality assurance are:

|  |  |
| --- | --- |
| **Management Responsibility** | **Quality System** |
| Control of non conforming products | Design Control |
| Handling, storage, packaging & delivery | Purchasing |
| Purchaser-supplied products | Product identification & traceability |
| Process Control | Inspection & Testing |
| Inspection & Test equipments | Inspection & test status |
| Contract Review | Corrective action |
| Document Control | Quality Records |
| Internal Quality Audits | Training |
| Servicing | Statistical Techniques |

* **Six Sigma (6σ) :-**

Six Sigma is a methodology for pursuing continuous improvement in customer satisfaction and profit. It is a management philosophy attempting to improve effectiveness and efficiency.

**Origin of Six Sigma:-**

* Six Sigma originated at Motorola in the early 1980s, in response to achieving 10X reduction in product-failure levels in 5 years.
* Engineer Bill Smith invented Six Sigma, but died of a heart attack in the Motorola cafeteria in 1993, never knowing the scope of the craze and controversy he had touched off.
* Six Sigma is based on various quality management theories (e.g. Deming's 14 point for management, Juran's 10 steps on achieving quality).

**Features of Six Sigma:-**

* Six Sigma's aim is to eliminate waste and inefficiency, thereby increasing customer satisfaction by delivering what the customer is expecting.
* Six Sigma follows a structured methodology, and has defined roles for the participants.
* Six Sigma is a data driven methodology, and requires accurate data collection for the processes being analyzed.
* Six Sigma is about putting results on Financial Statements.
* Six Sigma is a business-driven, multi-dimensional structured approach for −
  + Improving Processes
  + Lowering Defects
  + Reducing process variability
  + Reducing costs
  + Increasing customer satisfaction
  + Increased profits

*The word Sigma is a statistical term that measures how far a given process deviates from perfection.*

The central idea behind Six Sigma: If you can measure how many "defects" you have in a process, you can systematically figure out how to eliminate them and get as close to "zero defects" as possible and specifically it means a failure rate of 3.4 parts per million or 99.9997% perfect.

**Key Concepts of Six Sigma**

At its core, Six Sigma revolves around a few key concepts.

* **Critical to Quality** − Attributes most important to the customer.
* **Defect** − Failing to deliver what the customer wants.
* **Process Capability** − What your process can deliver.
* **Variation** − What the customer sees and feels.
* **Stable Operations** − Ensuring consistent, predictable processes to improve what the customer sees and feels.
* **Design for Six Sigma** − Designing to meet customer needs and process capability.

Our Customers Feel the Variance, Not the Mean. So Six Sigma focuses first on reducing process variation and then on improving the process capability.

**Myths about Six Sigma:-**

There are several myths and misunderstandings surrounding Six Sigma. Some of them few are given below −

* Six Sigma is only concerned with reducing defects.
* Six Sigma is a process for production or engineering.
* Six Sigma cannot be applied to engineering activities.
* Six Sigma uses difficult-to-understand statistics.
* Six Sigma is just training.

**Benefits of Six Sigma:-**

Six Sigma offers six major benefits that attract companies −

* Generates sustained success
* Sets a performance goal for everyone
* Enhances value to customers
* Accelerates the rate of improvement
* Promotes learning and cross-pollination
* Executes strategic change

**There are three key elements of Six Sigma Process Improvement −**

* Customers
* Processes
* Employees

**The Customers**

Customers define quality. They expect performance, reliability, competitive prices, on-time delivery, service, clear and correct transaction processing and more. This means it is important to provide what the customers need to gain customer delight.

**The Processes**

Defining processes as well as defining their metrics and measures is the central aspect of Six Sigma.

In a business, the quality should be looked from the customer's perspective and so we must look at a defined process from the outside-in.

By understanding the transaction lifecycle from the customer's needs and processes, we can discover what they are seeing and feeling. This gives a chance to identify weak areas within a process and then we can improve them.

**The Employees**

A company must involve all its employees in the Six Sigma program. Company must provide opportunities and incentives for employees to focus their talents and ability to satisfy customers.

It is important to Six Sigma that all the team members should have a well-defined role with measurable objectives.

**Six Sigma Start-up**

Now you have decided to go for Six Sigma. So what is next?

Deploying Six Sigma within an organization is a big step and involves many activities including define, measure, analyze, improve, and control phases. Here are some steps, which are required for an organization at the time of starting Six Sigma implementation.

* **Plan your own route** − There may be many paths to Six Sigma but the best is the one that works for your organization.
* **Define your objective** − It is important to decide what you want to achieve, and priorities are important.
* **Stick to what is feasible** − Set up your plans so that they can match your influences, resources and scope.
* **Preparing Leaders** − They are required to launch and guide the Six Sigma Effort.
* **Creating Six Sigma organization** − This includes preparing Black Belts and other roles and assigning them their responsibilities.
* **Training the organization** − Apart from having black belts, it is required to impart training of Six Sigma to all the employees in the organization.
* **Piloting Six Sigma effort** − Piloting can be applied to any aspect of Six Sigma including solutions derived from process improvement or design redesign projects.

**Project Selection for Six Sigma**

One of the most difficult challenges in Six Sigma is the selection of the most appropriate problem to attack. There are generally two ways to generate projects −

* **Top-down** − This approach is generally tied to business strategy and is aligned with customer needs. The major weakness is they are too broad in scope to be completed in a timely manner (most six sigma projects are expected to be completed in 3-6 months).
* **Bottom-up** − In this approach, Black Belts choose the projects that are well-suited for the capabilities of teams. A major drawback of this approach is that, projects may not be tied directly to strategic concerns of the management thereby, receiving little support and low recognition from the top.

**Six Sigma has two key methodologies −**

* **DMAIC** − It refers to a data-driven quality strategy for improving processes. This methodology is used to improve an existing business process.
* **DMADV** − It refers to a data-driven quality strategy for designing products & processes. This methodology is used to create new product designs or process designs in such a way that it results in a more predictable, mature and defect free performance.

There is one more methodology called **DFSS** − **D**esign **F**or **S**ix **S**igma. DFSS is a data-driven quality strategy for designing or redesigning a product or service from the ground up.

Sometimes a DMAIC project may turn into a DFSS project because the process in question requires complete redesign to bring about the desired degree of improvement.

**DMAIC Methodology**

This methodology consists of the following five steps.

**D**efine --> **M**easure --> **A**nalyze --> **I**mprove -->**C**ontrol

* **Define** − Define the problem or project goal that needs to be addressed.
* **Measure** − Measure the problem and process from which it was produced.
* **Analyze** − Analyze data and process to determine root causes of defects and opportunities.
* **Improve** − Improve the process by finding solutions to fix, diminish, and prevent future problems.
* **Control** − Implement, control, and sustain the improvements solutions to keep the process on the new course.

We will discuss more on DMAIC Methodology in the subsequent chapters.

**DMADV Methodology**

This methodology consists of five steps −

**D**efine --> **M**easure --> **A**nalyze --> **D**esign -->**V**erify

* **Define** − Define the Problem or Project Goal that needs to be addressed.
* **Measure** − Measure and determine customers needs and specifications.
* **Analyze** − Analyze the process to meet the customer needs.
* **Design** − Design a process that will meet customers needs.
* **Verify** − Verify the design performance and ability to meet customer needs.

**DFSS Methodology (D**esign **F**or **S**ix **S**igma):-

DFSS is a separate and emerging discipline related to Six Sigma quality processes. This is a systematic methodology utilizing tools, training, and measurements to enable us to design products and processes that meet customer expectations and can be produced at Six Sigma Quality levels.

This methodology can have the following five steps.

**D**efine --> **I**dentify --> **D**esign --> **O**ptimize -->**V**erify

* **Define** − Define what the customers want, or what they do not want.
* **Identify** − Identify the customer and the project.
* **Design** − Design a process that meets customers needs.
* **Optimize** − Determine process capability and optimize the design.
* **Verify** − Test, verify, and validate the design.
* **Six Sigma - Define Phase :-**

There are five high-level steps in the application of Six Sigma to improve the quality of output. The first step is Define. During the Define phase, four major tasks are undertaken.

**Project Team is Formation**

Perform two activities

* Determine who needs to be on the team.
* What roles will each person perform?

Picking the right team members can be a difficult decision, especially if a project involves a large number of departments. In such projects, it could be wise to break them down into smaller pieces and work toward completion of a series of phased projects.

**Document Customers Core Business Processes**

Every project has customers. A customer is the recipient of the product or service of the process, targeted for improvement. Every customer has one or multiple needs from his or her supplier. For each need provided for, there are requirements for the need. The requirements are the characteristics of the need that determine whether the customer is happy with the product or service provided. So, document customer needs and related requirements.

A set of business processes is documented. These processes will be executed to meet customer's requirements and to resolve their Critical to Quality issues.

**Develop a Project Charter**

This is a document that names the project, summarizes the project by explaining the business case in a brief statement, and lists the project scope and goals. A project charter has the following components −

* Project name
* Business case
* Project scope
* Project goals
* Milestones
* Special requirements
* Special assumptions
* Roles and responsibilities of the project team

**Develop the SIPOC process map**

A process is defined as a series of steps and activities that take inputs, add value, and produce an output.

SIPOC is a process map that identifies all the following elements of a project −

* Suppliers
* Input
* Process
* Output
* Customers

The SIPOC process map is essential for identifying −

* The way processes occur currently.
* How those processes should be modified and improved throughout the remaining phases of DMAIC.
* **Six Sigma - Measure Phase:-**

During the Measure Phase, the overall performance of the Core Business Process is measured.

There are three important parts of Measure Phase.

**Data Collection Plan and Data Collection**

A data collection plan is prepared to collect the required data. This plan includes what type of data needs to be collected, what are the sources of data, etc. The reason to collect data is to identify areas where current processes need to be improved.

You collect data from three primary sources: input, process, and output.

* The input source is where the process is generated.
* Process data refers to tests of efficiency: the time requirements, cost, value, defects or errors, and labor spent on the process.
* Output is a measurement of efficiency.

**Data Evaluation**

At this stage, the collected data is evaluated and sigma is calculated. It gives an approximate number of defects.

* A Six Sigma defect is defined as anything outside of customer specifications.
* A Six Sigma opportunity is the total quantity of chances for a defect.

First we calculate Defects Per Million Opportunities (DPMO), and based on that a Sigma is decided from a predefined table −

Number of defects

DPMO = -------------------------------------------------- x 1,000,000

Number of Units x Number of opportunities

As stated above, Number of defects is the total number of defects found, Number of Units is the number of units produced, and number of opportunities means the number of ways to generate defects.

For example, the food ordering delivery project team examines 50 deliveries and finds out the following −

* Delivery is not on time (13)
* Ordered food is not according to the order (3)
* Food is not fresh (0)

So now, DPMO will be as follows −

13 + 3

DPMO = ----------- x 1,000,000 = 106,666.7

50 x 3

According to the Yield to Sigma Conversion Table given, below 106,666.7 defects per million opportunities is equivalent to a sigma performance of between 2.7 and 2.8.

This is the method used for measuring results as we proceed through a project. This beginning point enables us to locate the cause and effect of those processes and to seek defect point so that the procedure can be improved.

**Failure Mode and Effects Analysis - FMEA**

The final segment of the measure phase is called FMEA. It refers to preventing defects before they occur. The FMEA process usually includes rating possible defects, or failures, in three ways:

* The likelihood that something might go wrong.
* The ability to detect a defect.
* The level of severity of the defect.

You may use a rating scale. For example, rate each of these three areas from 1 to 10, with 1 being the lowest FMEA level and 10 being the highest. The higher the level, the more severe the rating. Hence, a high FMEA indicates the need to devise and implement improved measuring steps within the overall process. This would have the effect of preventing defects. Clearly, there is no need to spend a lot of time on this procedure if the likelihood of a defect is low.

**Yield to Sigma Conversion Table**

|  |  |  |
| --- | --- | --- |
| **Yield %** | **Sigma** | **Defects Per Million Opportunities** |
| 99.9997 | 6.00 | 3.4 |
| 99.9995 | 5.92 | 5 |
| 99.9992 | 5.81 | 8 |
| 99.9990 | 5.76 | 10 |
| 99.9980 | 5.61 | 20 |
| 99.9970 | 5.51 | 30 |
| 99.9960 | 5.44 | 40 |
| 99.9930 | 5.31 | 70 |
| 99.9900 | 5.22 | 100 |
| 99.9850 | 5.12 | 150 |
| 99.9770 | 5.00 | 230 |
| 99.9670 | 4.91 | 330 |
| 99.9520 | 4.80 | 480 |
| 99.9320 | 4.70 | 680 |
| 99.9040 | 4.60 | 960 |
| 99.8650 | 4.50 | 1350 |
| 99.8140 | 4.40 | 1860 |
| 99.7450 | 4.30 | 2550 |
| 99.6540 | 4.20 | 3460 |
| 99.5340 | 4.10 | 4660 |
| 99.3790 | 4.00 | 6210 |
| 99.1810 | 3.90 | 8190 |
| 98.9300 | 3.80 | 10700 |
| 98.6100 | 3.70 | 13900 |
| 98.2200 | 3.60 | 17800 |
| 97.7300 | 3.50 | 22700 |
| 97.1300 | 3.40 | 28700 |
| 96.4100 | 3.30 | 35900 |
| 95.5400 | 3.20 | 44600 |
| 94.5200 | 3.10 | 54800 |
| 93.3200 | 3.00 | 66800 |
| 91.9200 | 2.90 | 80800 |
| 90.3200 | 2.80 | 96800 |
| 88.5000 | 2.70 | 115000 |
| 86.5000 | 2.60 | 135000 |
| 84.2000 | 2.50 | 158000 |
| 81.6000 | 2.40 | 184000 |
| 78.8000 | 2.30 | 212000 |
| 75.8000 | 2.20 | 242000 |
| 72.6000 | 2.10 | 274000 |
| 69.2000 | 2.00 | 308000 |
| 65.6000 | 1.90 | 344000 |
| 61.8000 | 1.80 | 382000 |
| 58.0000 | 1.70 | 420000 |
| 54.0000 | 1.60 | 460000 |
| 50.0000 | 1.50 | 500000 |
| 46.0000 | 1.40 | 540000 |
| 43.0000 | 1.32 | 570000 |
| 39.0000 | 1.22 | 610000 |
| 35.0000 | 1.11 | 650000 |
| 31.0000 | 1.00 | 690000 |
| 28.0000 | 0.92 | 720000 |
| 25.0000 | 0.83 | 750000 |
| 22.0000 | 0.73 | 780000 |
| 19.0000 | 0.62 | 810000 |
| 16.0000 | 0.51 | 840000 |
| 14.0000 | 0.42 | 860000 |
| 12.0000 | 0.33 | 880000 |
| 10.0000 | 0.22 | 900000 |
| 8.0000 | 0.09 | 920000 |

* **Six Sigma – Analyze Phase :-**

Six Sigma aims to define the causes of defects, measure those defects, and analyze them so that they can be reduced. We consider five specific types of analyses that help to promote the goals of the project. These are source, process, data, resource, and communication analysis. Now we will see them in detail.

**Source Analysis**

This is also called root cause analysis. It attempts to find defects that are derived from the sources of information or work generation. After finding the root cause of the problem, attempts are made to resolve the problem before we expect to eliminate defects from the product.

Three Steps to Root Cause Analysis

* The open step − During this phase, the project team brainstorms all the possible explanations for current sigma performance.
* The narrow step − During this phase, the project team narrows the list of possible explanations for current sigma performance.
* The close step − During this phase, the project team validates the narrowed list of explanations that explain sigma performance.

**Process Analysis**

Analyze the numbers to find out how well or poorly the processes are working, compared to what's possible and what the competition is doing.

Process analysis includes creating a more detailed process map, and analyzing the more detailed map, where the greatest inefficiencies exist.

The source analysis is often difficult to distinguish from process analysis. The process refers to the precise movement of materials, information, or requests from one place to another.

**Data Analysis**

Use of measures and data (those already collected or new data gathered in the analyze phase) to discern patterns, tendencies or other factors about the problem that either suggest or prove/disprove possible cause of the problem.

The data itself may have defect. There may be a case when products or deliverables do not provide all the needed information. Hence data is analyzed to find out defects and attempts are made to resolve the problem before we expect to eliminate defects from the product.

**Resource Analysis**

We also need to ensure that employees are properly trained in all departments that affect the process. If training is inadequate, you want to identify that as a cause of defects.

Other resources include raw materials needed to manufacture, process, and deliver the goods. For example, if the Accounting Department is not paying vendor bills on time and, consequently, the vendor holds up a shipment of shipping supplies, it becomes a resource problem.

**Communication analysis**

One problem common to most processes high in defects is poor communication. The classic interaction between a customer and a retail store is worth studying because many of the common communication problems are apparent in this case.

The same types of problems occur with internal customers as well, even though we may not recognize the sequence of events as a customer service problem.

The exercise of looking at issues from both points of view is instructive. A vendor wants payment according to agreed-upon terms, but the Accounting Department wants to make its batch processing uniform and efficient. Between these types of groups, such disconnects demonstrates the importance of communication analysis.

* **Six Sigma – Improve Phase :-**

If the project team does a thorough job in the root causation phase of analysis, the improve phase of DMAIC can be quick, easy, and satisfying work.

The objective of Improve Phase is to identify improvement breakthroughs, identify high gain alternatives, select preferred approach, design the future state, determine the new Sigma level, perform cost/benefit analysis, design dashboards/ scorecards, and create a preliminary implementation plan.

* Identify Improvement Breakthroughs −
  + Apply idea-generating tools and techniques to identify potential solutions that eliminate root causes.
* Identify/Select High Gain Alternatives −
  + Develop criteria to evaluate candidate improvement solutions.
  + Think systematically and holistically.
  + Prioritize and evaluate the candidate solutions against the solution evaluation criteria.
  + Conduct a feasibility assessment for the highest value solutions.
  + Develop preliminary solution timelines and cost-benefit analysis to aid in recommendation presentation and future implementation planning.

Improvement can involve a simple fix once we discover the causes of defects. However, in some cases, we may need to employ additional tools as well. These include −

* Solution alternatives
* Experiments with solution alternatives
* Planning for future change
* **Six Sigma – Control Phase :-**

The last phase of DMAIC is control, which is the phase where we ensure that the processes continues to work well, produce desired output results, and maintain quality levels. You will be concerned with four specific aspects of control, which are as follows.

**Quality control**

The ultimate purpose in control is the overall assurance that a high standard of quality is met. Customer's expectations depend on this, so control is inherently associated with quality.

Since the purpose of Six Sigma is to improve the overall process by reducing defects, quality control is the essential method for keeping the whole process on track; for enabling us to spot trouble and fix it; and for judging how effectively the project was executed and implemented.

Quality is at the heart of Six Sigma philosophy. Reducing defects has everything to do with striving for perfection. Whether we reach perfection or not, the effort defines our attitude toward quality itself.

**Standardization**

Standardization enables processes to go as smoothly as possible. In a manufacturing environment, the value of standardization has been proven over and over.

We need to devise a control feature to processes so that the majority of work is managed in a standardized manner.

**Control Methods and Alternatives**

The development of a new process of any change to an existing process requires the development of procedures to control work flow.

When a process cannot be managed in the normal manner, we need to come up with alternatives, short of forcing compliance to the standardized method.

**Responding when Defects Occur**

The final step in a control process is knowing how to respond once a defect is discovered. The weak links in the procedure, where defects are most likely to occur, can and should be monitored carefully so that defects can be spotted and fixed before the process continues.

The response to a defect may be to prevent a discovered flaw from becoming a defect at all. In the best designed systems, defects can be reduced to near zero, so that we may actually believe that Six Sigma can be attained.

**Six Sigma - Defect Metrics**

Before we go ahead, let us define two terms:

* A Six Sigma defect is defined as anything outside of customer specifications.
* A Six Sigma opportunity is the total quantity of chances for a defect.

This chapter provides a list formulae normally used to measure different metrics related to Six Sigma defects.

**Defects Per Unit - DPU**

Total Number of Defects

DPU = -----------------------------------

Total number of Product Units

The probability of getting 'r' defects in a sample having a given DPU rate can be predicted with the Poisson Distribution.

Total Opportunities - TO

TO = Total number of Product Units x Opportunities

**Defects Per Opportunity - DPO**

Total Number of Defects

DPO = -------------------------------

Total Opportunity

Defects Per Million Opportunities - DPMO

DPMO = DPO x 1,000,000

Defects Per Million Opportunities or DPMO can be then converted to sigma values using Yield to Sigma Conversion Table given in [Six Sigma - Measure Phase](http://www.tutorialspoint.com/six_sigma/six_sigma_measure_phase.htm).

According to the conversion table −

6 Sigma = 3.4 DPMO

How to find your Sigma Level

* Clearly define the customer's explicit requirements.
* Count the number of defects that occur.
* Determine the yield-percentage of items without defects.
* Use the conversion chart to determine DPMO and Sigma Level.

**Simplified Sigma Conversion Table**

|  |  |  |
| --- | --- | --- |
| If your yield is | Your DPMO is | Your Sigma is |
| 30.9% | 690,000 | 1.0 |
| 62.9% | 308,000 | 2.0 |
| 93.3 | 66,800 | 3.0 |
| 99.4 | 6,210 | 4.0 |
| 99.98 | 320 | 5.0 |
| 99.9997 | 3.4 | 6.0 |

* **1.3 SQA Activities:-**

There are two different groups involved in SQA related activities:

* Software engineers who do the technical work
* SQA group who is responsible for QA planning, oversight (Mistake or Failure to notice) , record keeping, analysis, and reporting

Software engineers address quality by applying solid technical methods and measures, conducting formal and technical reviews, and performing well planned software testing.

The SQA group assists the software team in achieving a high quality product.

**SQA Group Activities**

An SQA plan is developed for the project during project planning and is reviewed by all stake holders. The plan includes the identification of:

* Evaluations to be performed
* Audits and reviewed to be performed
* Standards that are applicable to the project
* Procedures for error reporting and tracking
* Documents to be produced by the SQA group
* Amount of feedback provided to the software project team

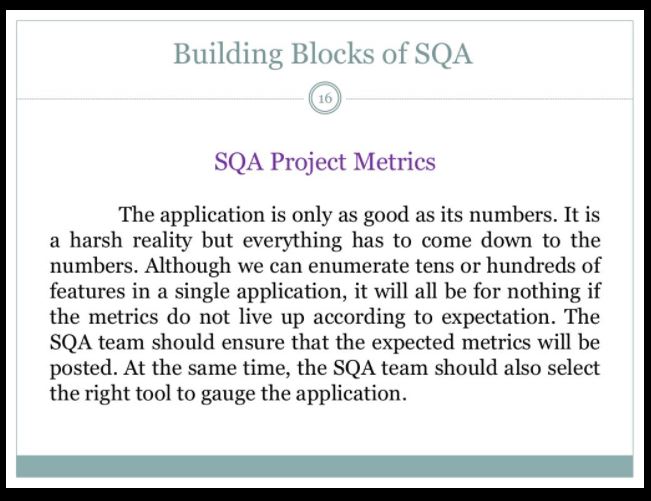
The group participates in the development of the project’s software process description. The software team selects the process and SQA group reviews the process description for compliance with the organizational policies, internal software standards, externally imposed standards, and other parts of the software project plan.

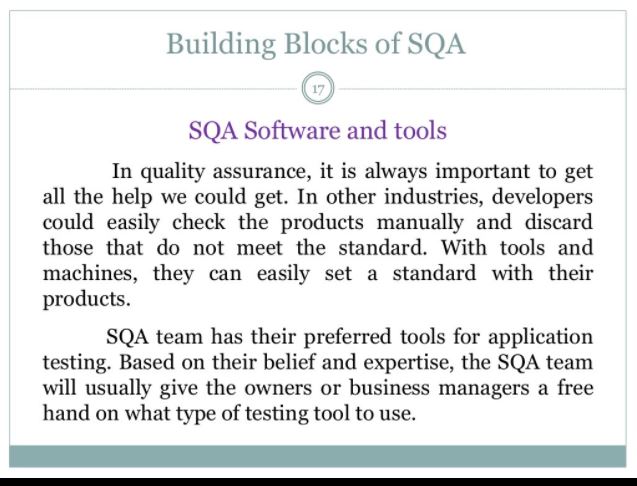
The SQA group also reviews software engineering activities to verify compliance with the defined software process. It identifies, documents, and tracks deviations from the process and verifies that the corrections have been made. In addition, it audits designated software work products to verify compliance with those defined as part of the software process. It, reviews selected work products, identifies, documents, and tracks deviations; verifies that corrections have been made; and reports the results of its work to the project manager.

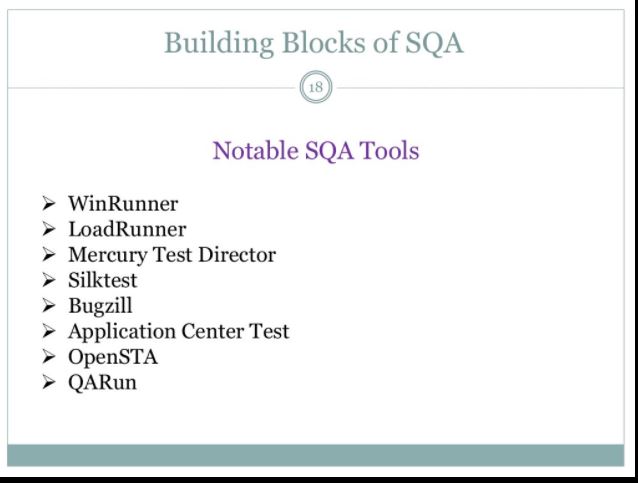
The basis purpose is to ensure that deviations in software work and work products are documented and handled according to documented procedures. These deviations may be encountered in the project plan, process description, applicable standards, or technical work products. The group records any non-compliance and reports to senior management and non-compliant items are recorded and tracked until they are resolved.

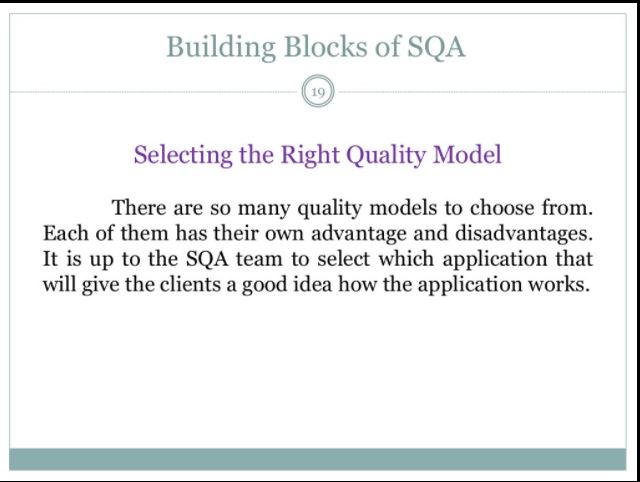
Another very important role of the group is to coordinate the control and management of change and help to collect and analyze software metrics.

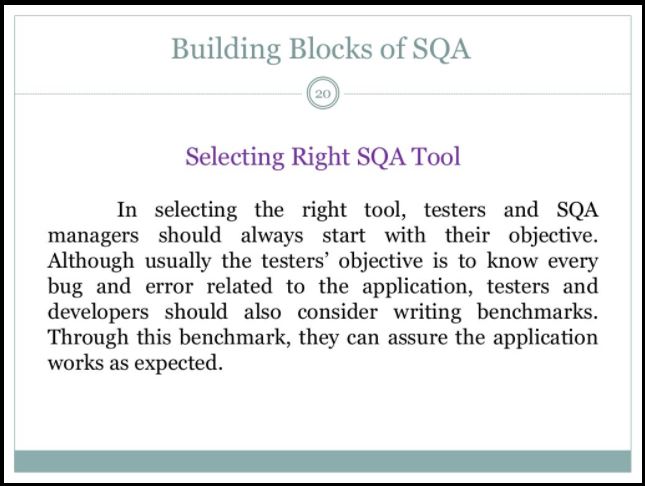
* **1.4 Building Blocks of SQA :-**

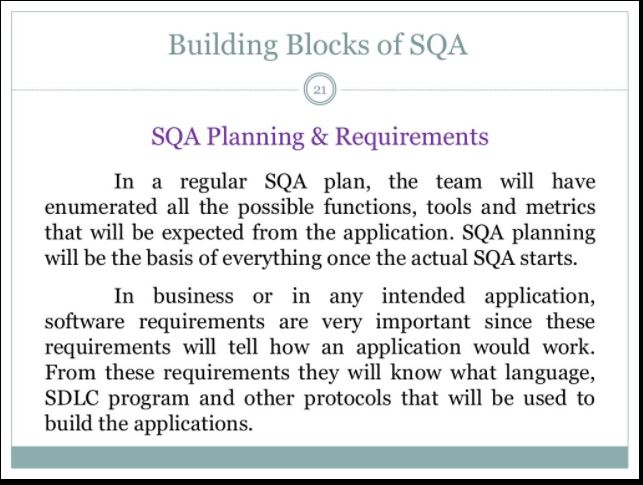
****

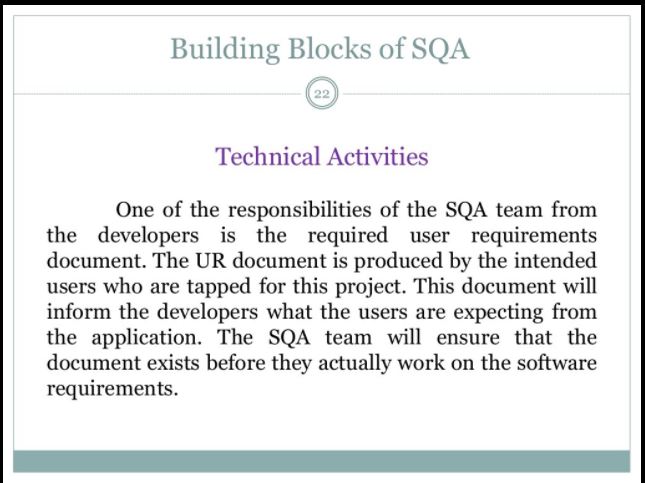
****

****

****

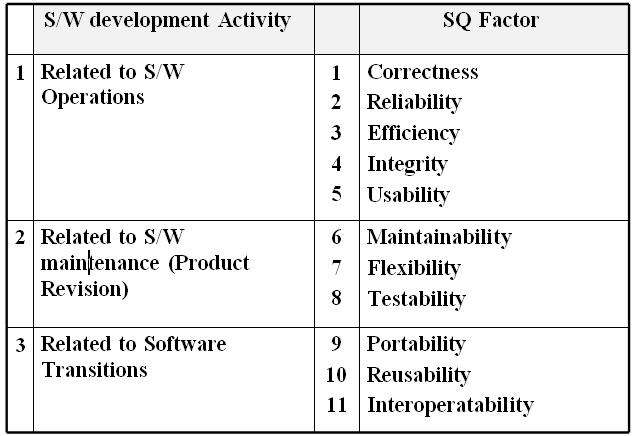
****

****

****

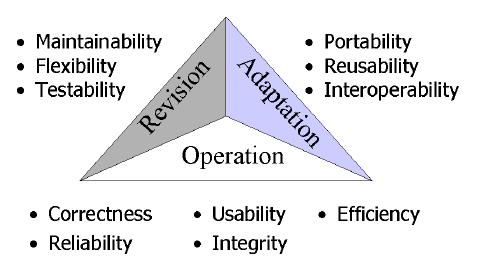
* **1.5 Software Quality factors:**

In 1978, McCall identified factors that could be used to develop metrics for the software quality. According to McCalls SQ factors are classified according to three major activities of software development:



**Software Quality Factors:**

* Correctness: Satisfies to customer with respect to objective.
* Reliability: the extent to which s/w provides correct result.
* Efficiency: Reflects the amount of required resources & code to perform the specified function.
* Integrity: Control on unauthorized access to s/w or data.
* Usability: the effort required to understand the system i.e. to learn, operate, prepare i/p & interpret o/p.
* Maintainability: the effort required to locate & fix bugs in s/w.
* Flexibility: the effort required to modify an operational program.
* Testability: the effort required to test the program to ensure that it performs its intended function.
* Portability: ability of transferring the program from one environment to another.
* Reusability: the extent to which a program (or part) can be reused in different applications.
* Interoperability: the effort required to couple one system with other.



**Figure : Software Quality Factor**

**Attributes of Software Quality Factor:**

1. Accuracy: The precision of computations & controls.

2. Communication Commonality: The degree to which communication standards are used with respect to interface, B.w., protocol

3. Completeness: Shows the degree to which full implementation of required function has been achieved.

4. Conciseness: compactness of program in terms of LOC.

5. Consistency: Use of uniform design & documentation techniques throughout the s/w development project.

6. Data Commonality: Use of standard data structure & types throughout the program.

7. Auditability: The ease with which the conformance to standards is checked.

8. Error Tolerance: The damage that occur with respect to an error in program

9. Execution Efficiency: The run time performance of a program.

10. Expandability: The degree to which the design, data, architecture can be extended.

11. Generality: The utility /span/breadth of program component.

12. H/W Independence: The degree to which s/w is decoupled from H/w.

13. Instrumentation: The degree to which the program monitors its own operation & identifies errors that do occur.

14. Modularity: The functional independence of program components.

15. Operability: The ease of operation of a program.

16. Security: Availability of mechanism that control & protect data.

17. Self Documentation: The degree to which source code provides meaningful documentation.

18. Simplicity: The ease with which program can be understood.

19. S/W System Independence: The degree to which, program is independent of non-standard programming features, OS characteristics, & other environment constraints.

20. Traceability: The ability to trace the design representation.

21. Training: The degree to which the s/w helps the new user to apply the system.

* **1.6 Software Quality Metrics: Process Metrics & Product Metrics :-**

Software metrics can be classified into three categories −

* **Product metrics** − Describes the characteristics of the product such as size, complexity, design features, performance, and quality level.
* **Process metrics** − These characteristics can be used to improve the development and maintenance activities of the software.
* **Project metrics** − This metrics describe the project characteristics and execution. Examples include the number of software developers, the staffing pattern over the life cycle of the software, cost, schedule, and productivity.

Some metrics belong to multiple categories. For example, the in-process quality metrics of a project are both process metrics and project metrics.

**Software quality metrics** are a subset of software metrics that focus on the quality aspects of the product, process, and project. These are more closely associated with process and product metrics than with project metrics.

Software quality metrics can be further divided into three categories −

* Product quality metrics
* In-process quality metrics
* Maintenance quality metrics

**Product Quality Metrics :-**

This metrics include the following −

* Mean Time to Failure
* Defect Density
* Customer Problems
* Customer Satisfaction

**Mean Time to Failure:-**

It is the time between failures. This metric is mostly used with safety critical systems such as the airline traffic control systems, avionics, and weapons.

**Defect Density:-**

It measures the defects relative to the software size expressed as lines of code or function point, etc. i.e., it measures code quality per unit. This metric is used in many commercial software systems.

**Customer Problems:-**

It measures the problems that customers encounter when using the product. It contains the customer’s perspective towards the problem space of the software, which includes the non-defect oriented problems together with the defect problems.

The problems metric is usually expressed in terms of **Problems per User-Month (PUM)**.

PUM = Total Problems that customers reported (true defect and non-defect oriented problems) for a time period + Total number of license months of the software during the period

Where,

Number of license-month of the software = Number of install license of the software × Number of months in the calculation period

PUM is usually calculated for each month after the software is released to the market, and also for monthly averages by year.

**Customer Satisfaction:-**

Customer satisfaction is often measured by customer survey data through the five-point scale −

* Very satisfied
* Satisfied
* Neutral
* Dissatisfied
* Very dissatisfied

Satisfaction with the overall quality of the product and its specific dimensions is usually obtained through various methods of customer surveys. Based on the five-point-scale data, several metrics with slight variations can be constructed and used, depending on the purpose of analysis. For example −

* Percent of completely satisfied customers
* Percent of satisfied customers
* Percent of dis-satisfied customers
* Percent of non-satisfied customers

Usually, this percent satisfaction is used.

**In-process Quality Metrics:-**

In-process quality metrics deals with the tracking of defect arrival during formal machine testing for some organizations. This metric includes −

* Defect density during machine testing
* Defect arrival pattern during machine testing
* Phase-based defect removal pattern
* Defect removal effectiveness

**Defect density during machine testing:-**

Defect rate during formal machine testing (testing after code is integrated into the system library) is correlated with the defect rate in the field. Higher defect rates found during testing is an indicator that the software has experienced higher error injection during its development process, unless the higher testing defect rate is due to an extraordinary testing effort.

This simple metric of defects per KLOC or function point is a good indicator of quality, while the software is still being tested. It is especially useful to monitor subsequent releases of a product in the same development organization.

**Defect arrival pattern during machine testing:-**

The overall defect density during testing will provide only the summary of the defects. The pattern of defect arrivals gives more information about different quality levels in the field. It includes the following −

* The defect arrivals or defects reported during the testing phase by time interval (e.g., week). Here all of which will not be valid defects.
* The pattern of valid defect arrivals when problem determination is done on the reported problems. This is the true defect pattern.
* The pattern of defect backlog overtime. This metric is needed because development organizations cannot investigate and fix all the reported problems immediately. This is a workload statement as well as a quality statement. If the defect backlog is large at the end of the development cycle and a lot of fixes have yet to be integrated into the system, the stability of the system (hence its quality) will be affected. Retesting (regression test) is needed to ensure that targeted product quality levels are reached.

**Phase-based defect removal pattern:-**

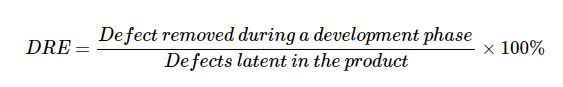
This is an extension of the defect density metric during testing. In addition to testing, it tracks the defects at all phases of the development cycle, including the design reviews, code inspections, and formal verifications before testing.

Because a large percentage of programming defects is related to design problems, conducting formal reviews, or functional verifications to enhance the defect removal capability of the process at the front-end reduces error in the software. The pattern of phase-based defect removal reflects the overall defect removal ability of the development process.

With regard to the metrics for the design and coding phases, in addition to defect rates, many development organizations use metrics such as inspection coverage and inspection effort for in-process quality management.

**Defect removal effectiveness:-**

It can be defined as follows −



This metric can be calculated for the entire development process, for the front-end before code integration and for each phase. It is called **early defect removal** when used for the front-end and **phase effectiveness** for specific phases. The higher the value of the metric, the more effective the development process and the fewer the defects passed to the next phase or to the field. This metric is a key concept of the defect removal model for software development.

**Maintenance Quality Metrics:-**

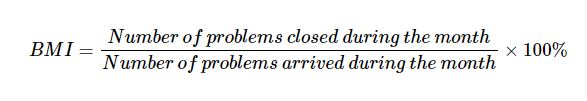
Although much cannot be done to alter the quality of the product during this phase, following are the fixes that can be carried out to eliminate the defects as soon as possible with excellent fix quality.

* Fix backlog and backlog management index
* Fix response time and fix responsiveness
* Percent delinquent fixes
* Fix quality

**Fix backlog and backlog management index:-**

Fix backlog is related to the rate of defect arrivals and the rate at which fixes for reported problems become available. It is a simple count of reported problems that remain at the end of each month or each week. Using it in the format of a trend chart, this metric can provide meaningful information for managing the maintenance process.

Backlog Management Index (BMI) is used to manage the backlog of open and unresolved problems.



If BMI is larger than 100, it means the backlog is reduced. If BMI is less than 100, then the backlog increased.

**Fix response time and fix responsiveness:-**

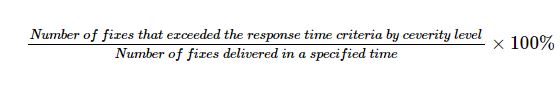
The fix response time metric is usually calculated as the mean time of all problems from open to close. Short fix response time leads to customer satisfaction.

The important elements of fix responsiveness are customer expectations, the agreed-to fix time, and the ability to meet one's commitment to the customer.

**Percent Delinquent Fixes :-**

It is calculated as follows –

Percent Delinquent Fixes =



**Fix Quality**

Fix quality or the number of defective fixes is another important quality metric for the maintenance phase. A fix is defective if it did not fix the reported problem, or if it fixed the original problem but injected a new defect. For mission-critical software, defective fixes are detrimental to customer satisfaction. The metric of percent defective fixes is the percentage of all fixes in a time interval that is defective.

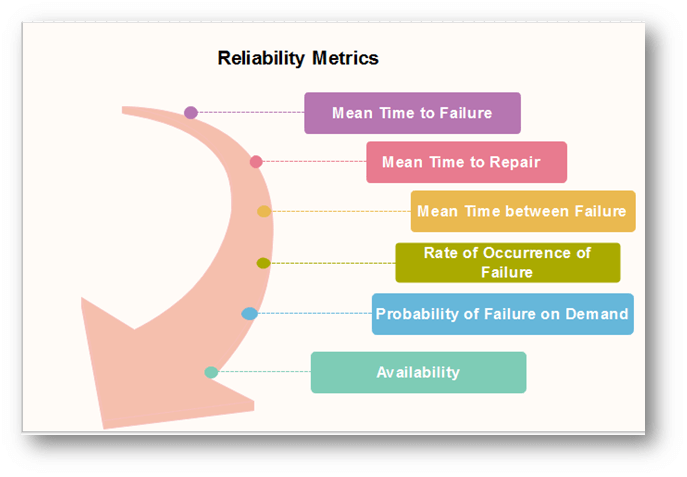
A defective fix can be recorded in two ways: Record it in the month it was discovered or record it in the month the fix was delivered. The first is a customer measure; the second is a process measure. The difference between the two dates is the latent period of the defective fix.

Usually the longer the latency, the more will be the customers that get affected. If the number of defects is large, then the small value of the percentage metric will show an optimistic picture. The quality goal for the maintenance process, of course, is zero defective fixes without delinquency.

* **1.7 Software Reliability & Reliability Measurement Factors:-**

Reliability metrics are used to quantitatively expressed the reliability of the software product. The option of which metric is to be used depends upon the type of system to which it applies & the requirements of the application domain.

Some reliability metrics which can be used to quantify the reliability of the software product are as follows:



1. **Mean Time to Failure (MTTF):-**

**MTTF** is described as the time interval between the two successive failures. An **MTTF** of 200 mean that one failure can be expected each 200-time units. The time units are entirely dependent on the system & it can even be stated in the number of transactions. **MTTF** is consistent for systems with large transactions.

For example, It is suitable for computer-aided design systems where a designer will work on a design for several hours as well as for Word-processor systems.

To measure **MTTF**, we can evidence the failure data for n failures. Let the failures appear at the time instants **t1,t2.....tn**.

**MTTF can be calculated as**

Reliability Metrics

2. **Mean Time to Repair (MTTR) :-**

Once failure occurs, some-time is required to fix the error. **MTTR** measures the average time it takes to track the errors causing the failure and to fix them.

3. **Mean Time Between Failure (MTBR) :-**

We can merge **MTTF** & **MTTR** metrics to get the MTBF metric.

**MTBF = MTTF + MTTR**

Thus, an **MTBF** of 300 denoted that once the failure appears, the next failure is expected to appear only after 300 hours. In this method, the time measurements are real-time & not the execution time as in **MTTF**.

4. **Rate of occurrence of failure (ROCOF):-**

It is the number of failures appearing in a unit time interval. The number of unexpected events over a specific time of operation. **ROCOF** is the frequency of occurrence with which unexpected role is likely to appear. A **ROCOF** of 0.02 mean that two failures are likely to occur in each 100 operational time unit steps. It is also called the failure intensity metric.

5. **Probability of Failure on Demand (POFOD):-**

**POFOD** is described as the probability that the system will fail when a service is requested. It is the number of system deficiency given several systems inputs.

**POFOD** is the possibility that the system will fail when a service request is made.

A **POFOD** of 0.1 means that one out of ten service requests may fail.**POFOD** is an essential measure for safety-critical systems. POFOD is relevant for protection systems where services are demanded occasionally.

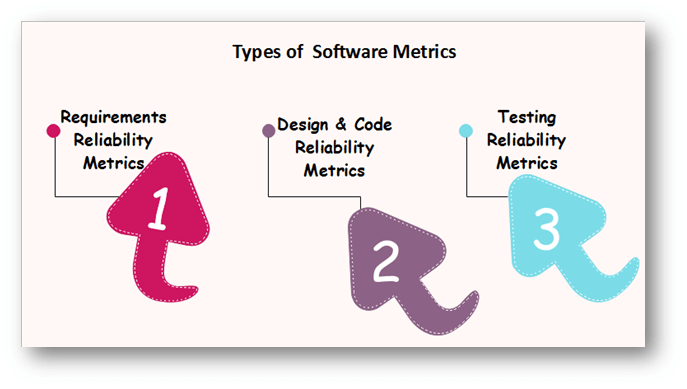
6. **Availability (AVAIL)**

Availability is the probability that the system is applicable for use at a given time. It takes into account the repair time & the restart time for the system. An availability of 0.995 means that in every 1000 time units, the system is feasible to be available for **995** of these. The percentage of time that a system is applicable for use, taking into account planned and unplanned downtime. If a system is down an average of four hours out of 100 hours of operation, its **AVAIL** is 96%.

**Software Metrics for Reliability**

The Metrics are used to improve the reliability of the system by identifying the areas of requirements.

**Different Types of Software Metrics are:-**



* **Requirements Reliability Metrics**

Requirements denote what features the software must include. It specifies the functionality that must be contained in the software. The requirements must be written such that is no misconception between the developer & the client. The requirements must include valid structure to avoid the loss of valuable data.

The requirements should be thorough and in a detailed manner so that it is simple for the design stage. The requirements should not include inadequate data. Requirement Reliability metrics calculates the above-said quality factors of the required document.

* **Design and Code Reliability Metrics**

The quality methods that exists in design and coding plan are complexity, size, and modularity. Complex modules are tough to understand & there is a high probability of occurring bugs. The reliability will reduce if modules have a combination of high complexity and large size or high complexity and small size. These metrics are also available to object-oriented code, but in this, additional metrics are required to evaluate the quality.

* **Testing Reliability Metrics**

These metrics use two methods to calculate reliability.

**First**, it provides that the system is equipped with the tasks that are specified in the requirements. Because of this, the bugs due to the lack of functionality reduces.

The **second** method is calculating the code, finding the bugs & fixing them. To ensure that the system includes the functionality specified, test plans are written that include multiple test cases. Each test method is based on one system state and tests some tasks that are based on an associated set of requirements. The goals of an effective verification program is to ensure that each elements is tested, the implication being that if the system passes the test, the requirement?s functionality is contained in the delivered system.

**Reliability Measures:**

* **What is reliability?**
* It is the probability of running a computer program without any failure in a specified environment for a specified time period.
* Failure means nonconformance to software requirements (as per software quality & reliability terms)
* A ***simple measure of reliability*** is *MTBF Mean Time Between Failure,* where

**MTBF = MTTF + MTTR**

* *MTTF: Mean Time To Failure, MTTR: Mean Time To Repair*
* Reliability is the most costly performance characteristic to assess & most difficult to guarantee.
* Software reliability is closely related with software availability.
* Availability is the percentage that a program is operating according to requirements at a given point of time.
* **Software Reliability** is the probability of a failure free operation over a specified time in a given environment for a specific purpose.
* **Software Availability** is the probability, that a system, at a point of time will be operational & able to deliver the requested services.
* Reliability can be improved by :

1. Fault avoidance

2. Fault Detection & Removal

3. Fault Tolerance

* **Software Reliability Metrics:**
* Metrics are used for specifying & measuring the software reliability & availability.

|  |  |  |
| --- | --- | --- |
|  | **Metric** | **Meaning** |
| **1** | **POFOD**  Probability of Failure on Demand | It is the probability that the system will fail when a service request is made. 0.001 POFOD means that out of 1000 service request 1 may result in failure. |
| **2** | **ROCOF**  Rate of Failure Occurrence | The frequency of the occurrence of unexpected behavior. This metric is sometimes called as Failure Intensity. 2/100 means 2 failure occurrence in each 100 operational time units. |
| **3** | **MTTF**  Mean Time To Failure | The average time between observed system failures. An MTTF of 500 means that one failure can be expected in every 500 time unit. |
| **4** | **AVAIL**  Availability | The probability that the system is available for use at a given time. Availability of 0.998 means that in ever 1000 time units the system is likely to be available for 998 of these. |

* **Software Reliability Measurements:**
* Three kinds of measurements can be made for assessing the system’s reliability :

**1. POFOD Measurement:**

The number of system failures with the given number of request for system services. This is used to measure the POFOD.

**2. ROCOF & MTTF Measurement:**

The time or the number of transactions between system failures. This is used to measure ROCOF & MTTF.

**3. AVAIL (Availability) Measurement:** The elapsed repair or restart time when a system failure occurs. This is used to measure the AVAIL (with assumption that the system must be continuously available).

* **Non –functional reliability Attributes are**
* Reliability
* Usability
* Maintainability
* Reusability (Adaptability w.r.t. other operating environment)
* Testability

**2.2 Reliability models:**

* Software reliability models are:

**1. Calendar time:**

The model that predicts reliability w.r.t. calendar time

**2. Processing (CPU) time:**

The model that predicts reliability w.r.t. processing (CPU) time (actual elapsed time in execution)

**3. Quantitative Model:**

The model based on the internal characteristics of a program & computes predicted no. of errors.

**4. Seeding Model:**

This model indicates the software reliability as a measure of the error detection power with a set of test cases.

* **Various criteria for reliability models are:**

1. **Predictive Validity:**

The ability of model of predicting the future failure of the system based on the collected test data.

1. **Capability:**

The ability of model to generate data that can be readily applied to s/w development efforts.

1. **Quality of assumptions:**

Assumptions on which the mathematical foundation of the model is based & the degree of

degradation of model when the assumption limits are over.

1. **Applicability:**

The degree to which the reliability model can be applied to different s/w application domains & types.

**Simplicity:**

It is the degree to which the collected data supports the model straightforward. It also refers as the degree to which the overall approach can be automated.