SOFTWARE QUALITY METRICS
PRODUCT QUALITY METRICS

1. *What is defect density in software quality? How to calculate defect density? Explain with suitable example.*

Software testing is an important phase in the software development. Testing is based on its quality, scalability, features, security, and performance, including other essential elements. It's common to detect defects and errors in a software testing process. To ensure the perfection of software, software engineers follow the defect density formula to determine the quality of the software.

In short we can say, more Defects then Lower Software Quality.

Defect density a mathematical calculation where output is the numerical data that determines the number of defects detected in software or component during a specific development period. In short, it is used to ensure whether the software is released or not.

By detecting defects and errors during the early stages of software development one can ensure the quality, performance, scalability, features, security, as well as other important elements of the software. Moreover, by conducting defect detection software developers can validate whether the application is being built as per the demands of the client and make all the necessary changes if required.

Definition of Defect Density : Defect Density is the number of defects confirmed in software/module during a specific period of operation or development divided by the size of the software/module. It enables one to decide if a piece of software is ready to be released.

The role of defect density is extremely important in Software Development Life Cycle (SDLC).

(1) It is used to identify the number of defects in software.

(2) It gives idea to the testing team to conduct additional efforts for re-engineering and replacements.

To Calculate Defect Density following formula is used :

Defect Density = Total Defect/Size

According to best practices, one defect per 1000 lines (LOC) is considered good. Such standard of defect density is called KLOC. The size of the software or code is expressed in Function Points (FP).

Steps to calculate Defect Density :

(1) Collect the total defects detected during the software development process.

(2) **Calculate Defect Density = Average number of Defects/KLOC**

Let's understand it with an example :

Example 1 :

For example: For a particular test cycl there are 30 defects in 5 modules or components. Therefore, the density will be:

Total no. of defects/ KLOC: 30/15 = 0.5

Hence, the density is 1 defect for every 2 KLOC

Example 2 :

Let's say your software comes with five integrated modules.

(1) *Module 1 = 5 bugs* **(2)** Module 2= 10 bugs

(3) Module 3= 20 bugs **(4)** Module 4= 15 bugs

(5) Module 5= 5 bugs

(6) Total bugs = 5+10+20+15+5= 55

Now total line of code for each module is

(1) Module 1= 500 LOC **(2)** Module 2= 1000 LOC

(3) Module 3= 1500 LOC **(4)** Module 4= 1500 LOC

(5) Module 5= 1000 LOC

Total Line of Code = 500 + 1000 + 1500 + 1500 + 1000 = 5500

Defect Density = 55/5500 = 0.01 defects/LOC or 10 defects/KLOC

2. *What are the different uses of Defect Density? Also state advantages.*

(1) Defect density is very important factor according to the industry standard for software and its compone development. Because it calculates the number o defects during development process and allow developers to determine the weak areas that requir robust testing.

(2) Hence organizations prefer defect density to release a product subsequently and compare them in terms of various aspects such as performance, security, quality, scalability, etc.

(3) Once defects are calculated using defect density formula, developers start to make changes accordingly to reduce those defects. The defect density process helps developers to determine how a reduction affects the software quality-wise.

(4) The use of defect density is important in many ways. However, once developers set up common defects, they can use this model to predict the remaining defects. Using this method, developers can establish a database of common defect densities to determine the productivity and quality of the product.

Advantages of Defect Density : Defect density having important role in software development. For software testers and developers, it provides details about defects.

(1) Developers can ensure that the product set to launch doesn't require any more testing

(2) Developers and testers can estimate the testing and rework required to fix the errors

(3) Testers can trace and detect components possessing high risks

(4) The testing team can determine the amount of training requires to complete the testing process

(5) One can identify the area of improvement and fix it.

3. *Explain Customer Problem metric in detail. How it is related with Customer Satisfaction Metrics?*

In the software development, it's always important to understand the need of customer as well as problems of the customer. Generally customer problems encountered when customer start using the software actually. 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.

There is now a higher than ever level of awareness of this subject, and people noticed that investment in improving customer experience is eventually a low-cost opportunity to generate high income.

(1) **Net Promoter Score (NPS)** : Net Promoter Score (NPS) helps to measure loyalty by asking your users, whether they are willing to recommend your product. It is measured from responses to the question – Considering your complete experience with our organization, how likely are you to recommend our products and services to your loved ones and colleagues.

(2) **Customer Service Satisfaction (CSS)** : CSS measures your customers' satisfaction after purchasing your product or service. You can measure CSS by seeking feedback from your customers every time they interact with your business.

(3) **Customer Effort Score (CES)** : Some products or services may be complex to operate because it requires large amount of efforts from the customers. Customer Effort Score (CES) helps determine ease of using your products or services.

(4) **Customer Satisfaction Score (CSAT)** : Customer Satisfaction Score lets you know if your customers are happy or not. It measures satisfaction using a rating scale question that asks survey takers to rate their satisfaction level with their product or service.

(5) **Customer Reviews** : While these cover all the angles, it is important to be mindful of any feedback or review you may receive via portals, websites, or social media channels.

4. *Explain function points in detail. Also state objectives, advantages and disadvantages of FP.*

It is important to express the amount of business functionality. A Function Point (FP) is a unit of measurement to express the amount of business functionality and an information system (as a product) provides to a user. They are widely accepted as an industry standard for functional sizing.

There is Object Management Group (OMG) which is an open membership computer industry standards consortium. It has adopted the Automated Function Point

(AFP) specification to maintain Software Quality. It provides a standard for automating FP counting according to the guidelines of the International Function Point User Group.

There are 2 techniques as follows :

(1) **Function Point Analysis (FPA) Technique :** It quantifies the functions contained within software which is meaningful to the software users.

(2) **Function Points (FP) Counting :** It considers a standard set of rules, processes and guidelines as defined by the International Function Point Users Group (IFPUG).

FPA provides a standard method to calculate size the software work product. This work product is the output of software new development. It measures functionality from the user's point of view i.e. how user can use the software and getting the benefit. Function Point Analysis (FPA) is a method or set of rules of to calculate Functional Size Measurement.

Objectives of FPA :

(1) To measure the functionality that the user requests and receives.

(2) To measure software development and maintenance.

(3) To minimize the overhead of the measurement process.

(4) To remain consistent measure among various projects and organizations.

Advantages of FPA :

(1) FPA is a tool to determine the size of functions provided by software.

(2) It is a tool to help users discover the benefit of software to their organization by counting functions that specifically match their requirements.

(3) It is a tool to measure software quality and productivity analysis.

(4) It is a tool to estimate the cost and resources required for software development and maintenance.

Disadvantages of FPA :

(1) Costly and expensive models.

(2) Less research data on function points.

(3) Need to create the design specification.

(4) Very time-consuming method.

5. What are the different In-process Quality Metrics?

There are 3 In-process Quality Metrics as follows :

(1) Defect Arrival Pattern,

(2) Phase-Based Defect Removal Pattern,

(3) Defect Removal Effectiveness

Defect rate can be calculated during formal machine testing. Formal machine testing is performed after code is integrated into the system library. Defect rate is correlated with all the defects in the field.

Higher defect rates indicate the higher rate of failure. It 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.

Overall defect density during testing is a summary indicator. The pattern of defect arrivals (or for that matter, times between failures) gives more information. Even with the same overall defect rate during testing, different patterns of defect arrivals indicate different quality levels in the field.

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 : The total defect density which will be performed during testing will provide only the summary of the defects. It does not provide the arrival pattern of defects. The pattern of defect arrivals gives more information about different quality levels in the field. It includes the following :

The pattern of various defects, how they can arrive or defects reported during the testing phase by time interval (e.g., week).

This pattern of valid defect arrivals helps to predict how defects can be arrived. This metric is important 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 can be said an extension of the defect density metric during testing. Additionally it tracks the defects at all phases of the development cycle, including the design reviews, code inspections, and formal verifications before testing.

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 reduce 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 :

$$DRE = \frac{\text{Defect removed during a development phase}}{\text{Defects latent in the product}} \times 100\%$$

Using this formula, Defect Removal Effectiveness can be calculated. This metric can be calculated for the entire development process. This process should be done 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.

6. *What are the different metrics to count Software Maintenance?*

Maintenance is the important aspect of Software Development process. It can be considered as critical factor to provide effective maintenance of equipment. It should be ensured that to provide quality operations timely with a minimal cost. However, those in the maintenance field understand that equipment reliability does not come easy.

Organizations need to set quality benchmarks to measure the current effectiveness and predict future

performance and use the data obtained to understand where to make improvements.

Maintenance Metrics : There are two categories of maintenance key performance indicators which include the leading and lagging indicators. The leading indicators signal future events and the lagging indicators follow the past events.

The leading indicator comprises from metrics like the Estimated vs actual performance and PM Compliance, while the lagging indicator are reflected in maintenance metrics like the Mean Time To Repair (MTTR) , Overall Equipment Effectiveness OEE and Mean time between failure (MTBF).

Using these maintenance metrics and turning the data into actionable information, organizations can acquire both qualitative and quantitative insights.

And there is no better way to spot opportunities for improvement. Here are some important maintenance metrics you should track if you want to improve and optimize your maintenance operations.

(1) **Planned Maintenance Percentage (PPC)** : This metrics represents the percentage of time spent on planned maintenance activities against the unplanned.

In simpler terms, this metric tells you how much maintenance work done on a particular asset was a part of your preventive maintenance plan versus how much time you've spent repairing it because it unexpectedly broke down.

In a great system, 90% of the maintenance should be planned.

The calculation is as follows:

PPC= (scheduled maintenance time/total maintenance hours) * 100

(2) **Overall Equipment Effectiveness (OEE)** : OEE is the measure of the productivity of a piece of equipment. It gives informed data on how effective organization's maintenance processes is running based on factors like equipment quality, performance, and availability.

A 100% OEE means that your system is producing no defects, as fast as possible, and with no stops in the production.

To calculate the OEE, you multiply the availability by the performance and quality :

OEE = availability * performance * quality

(3) **Mean Time To Repair (MTTR) :** MTTR is the measure of the repairable items maintainability.

The MTTR clock starts ticking when the repairs start and it goes on until operations are restored. This includes repair time, testing period, and return to the normal operating condition.

The goal of every organization is to reduce MTTR as much as possible. This is especially important for critical assets as ever additional hour you need to restore an asset to a working condition amount to huge losses for your firm.

To calculate MTTR, you divide the downtime period by the total number of downtimes:

MTTR= (SUM of downtime periods/ total number of repairs)

(4) **Mean Time Between Failures (MTBF) :** MTBF is the measure of the predicted time between one breakdowns to the next during normal operation.

In essence, MTBF tells you the expected lifetime for a specific piece of equipment. Higher MTBF means that the part (or product) you bought will work longer before it experiences failure.

To calculate the MTBF, you divide the total operational time by the number of failures:

MTBF= (SUM of operational time/total number of failures)

(5) **Preventive Maintenance Compliance (PMC) :** PM compliance is defined as the percentage of the preventive work scheduled and completed in a set time.

For example, you might have 60 Work Orders (that are a part of the PM plan) scheduled but 51 completed at the end of the month.

In this case:

PMC= (51/60) x 100 = 85%

This tells you that 85% of all preventive WO's have been covered for selected month.

7. ***Write a short note on Backlog Management Index (BMI).***

(1) Backlog Management Index is one of the important metrics which is closely monitored in Steady State of Maintenance and Support Projects.

(2) In most cases Backlog Management Index provides a very good indication of the success of the project work and an understanding of the stability and control.

(3) There is a tendency of BMI to decrease steadily with time in Steady State of Development and Support Projects without adequate controls, processes, checks and balances. The factors affecting BMI may be classified into operational and process related.

(4) Backlog Management Index (BMI) is an important metric to manage the backlog of open, unresolved problems in a Software Project.

(5) It is typically used in maintenance and Support projects (or phases) to ensure the project remains under control. It is related to both the rate of defect arrivals and the rate with which the fixes to the defects become available.

(6) A Backlog of workload is defined as the count of problems that remain unresolved after a particular time span (week/month).
*BMI= Number of Problems/Service Desk Incidents closed in the month * 100 %*

(7) If the number of Problems/Service Desk Incidents closed in the month is more than the number of arrivals, the BMI will be more than 100 percent.

(8) This will indicate that the Software system and the project for maintaining and supporting it are under control and moving towards more stability.

(9) In the long term if this condition continues, an intervention from Project Manager, Sponsors, Technical Resources and other Stakeholders may be necessary.

(10) The analysis of the BMI Trend is an important aspect of Software Quality Management, Project Management, and Continuous Service Improvements Initiatives which are critical in defining the success or failure of the Project Effort.

(11) This whitepaper focuses on identifying the BMI Influencers especially in Steady State of Maintenance and Support Projects. It will then focus on minimizing the negative influence of the factors and suggesting ways to prevent their recurrence.

8. *Explain Fix Response Time and Fix Quality in detail.*

Fix Response Time

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

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

It is calculated as follows :

$$\text{percent Delinquent Fixes} = \frac{\text{number of fixes that exceeded the response time criteria by ceverity level}}{\text{Number of fixes delivered in a specified time}} \times 100\%$$

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.

Quality metrics demonstrate how the organization is performing on waste reduction, quality control, and responsiveness, as well as other measures that matter to anyone making, using, or investing in your products.

Just like performance, quality metrics can change minute by minute (or second by second). That's why it's important for quality management practices to enable real-time visibility. With instant views of quality metrics production staff and plant managers can make the right

decisions in the moment. Likewise, analysts and executives can use the same data to steer the organization toward positive strategic outcomes. Quality metrics are actionable at every level-if they're complete, consistent, and accessible. Complete data means you have everything you need to answer questions and solve problems.

Consistent data is collected and presented in the same way, no matter where it's collected. Consistency enables comparative analyses-and accurate conclusions.

Accessible data is easy for the people to retrieve, whenever and however they need it-including remotely. The cloud enables companies to centralize their quality control metrics to make them widely available.

9. *What are the different Software Quality Indicators?*

Following are the software quality indicators :

Reliability : Reliability refers to the level of risk inherent in a software product, and the likelihood it will fail. It also addresses "stability," as termed by ISO: how likely are there to be regressions in the-software when changes are made.

A related term coined in recent years is "resilience." This views the problem from a different direction, asking what is the software's ability to deal with failure, which will inevitably happen. For example, modern applications based on containerized micro-services can be easily and automatically redeployed in case of failure, making them highly resilient.

By analyzing the number of defects, you will see how well the software will work and how long the system will run smoothly without crashing. If you want to have a robust codebase, low defect count is especially important.

Maintainability : Your software should have good maintenance. If there is any problem occurred after deployment, it should be removed and there should be facility of maintenance.

Testability : How well does the product support the testing? It depends on how well you control, automate, isolate, and observe the testing process.

Portability : It shows if you can use the software in various environments. There are no tools that can show the project's portability, but there are some means to do it. The

best option is to test the code on various platforms without waiting until the end of the development.

Reusability : To check if you can use the assets such as code again, use metrics that measure the number of interdependencies. The reusability depends on the availability of the modularity or loose coupling.

Performance : Performance is especially important in fields like algorithmic or transactional processing, where massive amounts of data need to be processed very quickly, and even small latency can cause significant problems. But today performance is becoming universally important as users of web and mobile applications demand high performance and become quickly frustrated if a system does not respond quickly.

(1) **Load Testing :** Conducted to understand the behavior of the system under a certain load, for example, with 1,000 concurrent users.

(2) **Stress Testing :** Understanding the upper limit of capacity of the system.

(3) **Soak Testing :** Checking if the system can handle a certain load for a prolonged period of time, and when performance starts to degrade.

(4) **Application Performance Monitoring (APM)** This is a new category of software that can provide detailed metrics of performance from the user's perspective.

Rate of Delivery : In agile development environments new iterations of software are delivered to users quickly. Many organizations today ship new versions of their software every week, every day, or even several times a day. This is known as Continuous Delivery, or in its extreme form, Continuous Deployment, in which every change to the software is immediately shipped to production.

Rate of software delivery is related to quality because a new version of a software system will typically contain improvements that can impact the user. A higher frequency of releases that are delivered to the user should, in theory, mean that the user gets better software faster.

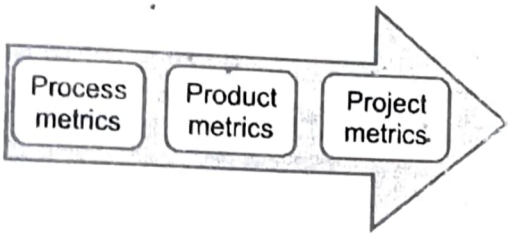
10. *What are the different Software Testing Metrics?*

Software Testing Metrics are the quantitative measures used to estimate the progress, quality

productivity and health of the software testing process. The goal of software testing metrics is to improve the efficiency and effectiveness in the software testing process and to help make better decisions for further testing process by providing reliable data about the testing process.

A Metric defines in quantitative terms the degree to which a system, system component, or process possesses a given attribute. The ideal example to understand metrics would be a weekly mileage of a car compared to its ideal mileage recommended by the manufacturer.

Types of Test Metrics :



(Figure)

(1) **Process Metrics :** It can be used to improve the process efficiency of the SDLC (Software Development Life Cycle)

(2) **Product Metrics :** It deals with the quality of the software product

(3) **Project Metrics :** It can be used to measure the efficiency of a project team or any testing tools being used by the team members

Identification of correct testing metrics is very important. Few things need to be considered before identifying the test metrics

(1) Fix the target audience for the metric preparation

(2) Define the goal for metrics

(3) Introduce all the relevant metrics based on project needs

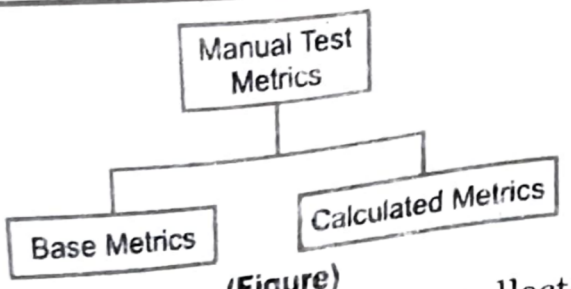
(4) Analyze the cost benefits aspect of each metrics and the project lifestyle phase in which it results in the maximum output.

11. *What are Manual Test Matrices?*

Software Engineering, Manual test metrics are classified into two classes

(1) Base Metrics

(2) Calculated Metrics



(Figure)

Base metrics is the raw data collected by Test Analyst during the test case development and execution (# of test cases executed, # of test cases). While calculated metrics are derived from the data collected in base metrics. Calculated metrics is usually followed by the test manager for test reporting purpose (% Complete, % Test Coverage).

Depending on the project or business model some of the important metrics are

(1) Test case execution productivity metrics
(2) Test case preparation productivity metrics
(3) Defect metrics
(4) Defects by priority
(5) Defects by severity
(6) Defect slippage ratio

Test Metrics Life Cycle :

Different Stages of Metrics Life Cycle	Steps During Each Stage
Analysis	Identification of the Metrics Define the identified QA Metrics
Communicate	Explain the need for metric to stakeholder and testing team Educate the testing team about the data points to need to be captured for processing the metric
Evaluation	Capture and verify the data Calculating the metrics value using the data captured
Report	Develop the report with an effective conclusion Distribute the report to the stakeholder and respective representative Take feedback from stakeholder