Unit-4

Dimension of Software Quality | Garvin's Dimensions Of Quality

David Garvin suggests that quality ought to be thought-about by taking a third-dimensional read point that begins with an assessment of correspondence and terminates with a transcendental (aesthetic) view. Though Garvin's 8 dimensions of quality weren't developed specifically for the software system, they'll be applied once software system quality is taken into account.

Eight dimensions of product quality management will be used at a strategic level to investigate quality characteristics. The idea was outlined by David A. Garvin, formerly C. Roland Christensen academician of Business Administration at Harvard grad school (died thirty Gregorian calendar month 2017).

A number of the scale are reciprocally reinforcing, whereas others don't seem to be, improvement in one is also at the expense of others. Understanding the trade-offs desired by customers among these dimensions will facilitate build a competitive advantage.

Garvin's eight dimensions will be summarized as follows:

1. **Performance Quality:**
   Will the software system deliver all content, functions, and options that are such as a part of the necessities model during a method that gives worth to the tip user?

2. **Feature Quality:**
   Does the software system offer options that surprise and delight first-time finish users?

3. **Reliability:**
   Will the software system deliver all options and capability while not failure?
   Is it obtainable once it's needed?
   Will it deliver practicality that's error-free?

4. **Conformance(fact of the rules):**
   Will the software system adjust to native and external software standards that are relevant to the application?
   Will it conform to the factual style and writing conventions? as an example, will the computer program conform to accepted style rules for menu choice or knowledge input?

5. **Durability:**
   Will the software system be maintained (changed) or corrected (debugged) while not the accidental generation of unintentional facet effects? can changes cause the error rate or responsibility to degrade with time?

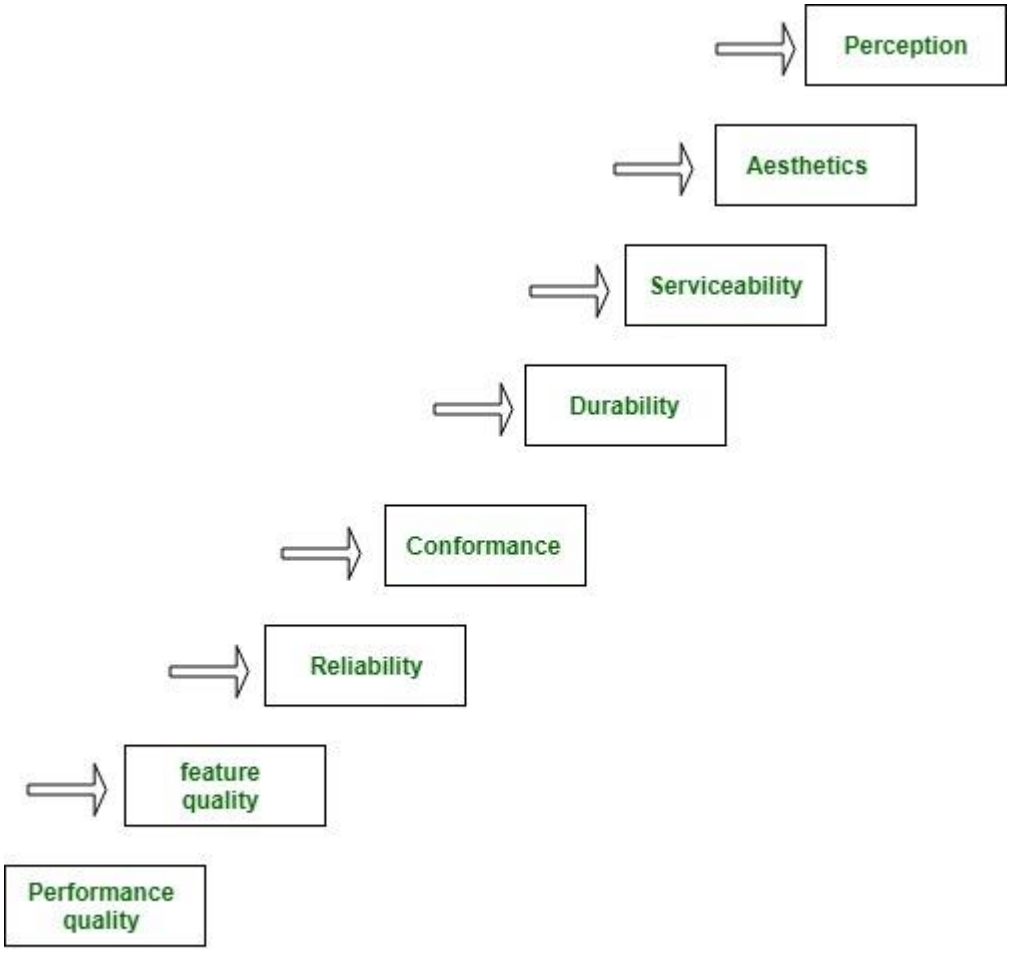
6. **Serviceability:**
   Will the software system be maintained (changed) or corrected (debugged) in a tolerably short time period?
   Will support employees acquire all data they have to create changes or correct defects?

7. **Aesthetics:**
   There's no doubt that every folk includes a totally different and really subjective vision of what's aesthetic.
   And yet, most folks would agree that an aesthetic entity includes a sure class, a novel flow, and a clear "presence" that are arduous to quantify however are evident still. The aesthetic software system has these characteristics.

8. **Perception:**
   In some things, you've got a collection of prejudices which will influence your perception of quality. as an example, if you're introduced to a software product that was engineered by a seller United Nations agency has created poor quality within the past, your guard is raised and your perception of the present software product quality may be influenced negatively.

Similarly, if a seller has a wonderful name, you will understand quality, even once it doesn't very exist.

Garvin's eight dimensions of quality in the diagrammatically form:



Garvin's Dimentions of Quality

Garvin's quality dimensions offer you with a "soft" take a look at software system quality. several (but not all) of those dimensions will solely be thought-about subjectively. For this reason, you furthermore might want a collection of "hard" quality factors that may be classified in 2 broad groups:

- Factors that can be directly measured (e.g., defects uncovered during testing).
- Factors that may be measured solely indirectly (e.g., usability or maintainability).
  In every case, activity should occur. you ought to compare the software system to some information and attain a sign of quality.

SOFTWARE QUALITY DIMENSIONS

Software Quality has many dimensions and below are some of them:

- **Accessibility:** The degree to which software can be used comfortably by a wide variety of people, including those who require assistive technologies like screen magnifiers or voice recognition.
- **Compatibility:** The suitability of software for use in different environments like different Operating Systems, Browsers, etc.
- **Concurrency:** The ability of software to service multiple requests to the same resources at the same time.
- **Efficiency:** The ability of software to perform well or achieve a result without wasted energy, resources, effort, time or money.
- **Functionality**: The ability of software to carry out the functions as specified or desired.
- **Installability:** The ability of software to be installed in a specified environment.
- **Localizability:** The ability of software to be used in different languages, time zones etc.
- **Maintainability:** The ease with which software can be modified (adding features, enhancing features, fixing bugs, etc)
- **Performance:** The speed at which software performs under a particular load.
- **Portability:** The ability of software to be transferred easily from one location to another.
- **Reliability:** The ability of software to perform a required function under stated conditions for stated period of time without any errors.
- **Scalability:** The measure of software's ability to increase or decrease in performance in response to changes in software's processing demands.
- **Security:** The extent of protection of software against unauthorized access, invasion of privacy, theft, loss of data, etc.
- **Testability:** The ability of software to be easily tested.
- **Usability:** The degree of software's ease of use.

Formal Technical Review (FTR) in Software Engineering

Formal Technical Review (FTR) is a software quality control activity performed by software engineers.

Objectives of formal technical review (FTR): Some of these are:

- Useful to uncover error in logic, function and implementation for any representation of the software.
- The purpose of FTR is to verify that the software meets specified requirements.
- To ensure that software is represented according to predefined standards.
- It helps to review the uniformity in software that is development in a uniform manner.
- To makes the project more manageable.

In addition, the purpose of FTR is to enable junior engineer to observe the analysis, design, coding and testing approach more closely. FTR also works to promote back up and continuity become familiar with parts of software they might not have seen otherwise. Actually, FTR is a class of reviews that include walkthroughs, inspections, round robin reviews and other small group technical assessments of software. Each FTR is conducted as meeting and is considered successful only if it is properly planned, controlled and attended.

Example:

suppose during the development of the software **without FTR** design cost 10 units, coding cost 15 units and testing cost 10 units then the total cost till now is 35 units without maintenance but there was a quality issue because of bad design so to fix it we have to re design the software and final cost will become 70 units. that is why FTR is so helpful while developing the software.

The review meeting: Each review meeting should be held considering the following constraints- *Involvement of people*:

1. Between 3, 4 and 5 people should be involve in the review.

2. Advance preparation should occur but it should be very short that is at the most 2 hours of work for every person.
3. The short duration of the review meeting should be less than two hour. Gives these constraints, it should be clear that an FTR focuses on specific (and small) part of the overall software.

   At the end of the review, all attendees of FTR must decide what to do.

1. Accept the product without any modification.
2. Reject the project due to serious error (Once corrected, another app need to be reviewed), or
3. Accept the product provisional (minor errors are encountered and should be corrected, but no additional review will be required).

   The decision was made, with all FTR attendees completing a sign-of indicating their participation in the review and their agreement with the findings of the review team.

   Review reporting and record keeping :-

1. During the FTR, the reviewer actively records all issues that have been raised.
2. At the end of the meeting all these issues raised are consolidated and a review list is prepared.
3. Finally, a formal technical review summary report is prepared.

   It answers three questions :-

1. What was reviewed ?
2. Who reviewed it ?
3. What were the findings and conclusions ?

   Review guidelines :- Guidelines for the conducting of formal technical reviews should be established in advance. These guidelines must be distributed to all reviewers, agreed upon, and then followed. A review that is unregistered can often be worse than a review that does not minimum set of guidelines for FTR.

1. Review the product, not the manufacture (producer).

2. Take written notes (record purpose)
3. Limit the number of participants and insists upon advance preparation.
4. Develop a checklist for each product that is likely to be reviewed.
5. Allocate resources and time schedule for FTRs in order to maintain time schedule.
6. Conduct meaningful training for all reviewers in order to make reviews effective.
7. Reviews earlier reviews which serve as the base for the current review being conducted.
8. Set an agenda and maintain it.
9. Separate the problem areas, but do not attempt to solve every problem notes.
10. Limit debate and rebuttal.

Software Engineering | Software Quality Assurance

Software Quality Assurance (SQA) is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly.

Software Quality Assurance is a process which works parallel to development of software. It focuses on improving the process of development of software so that problems can be prevented before they become a major issue. Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process.

Generally the quality of the software is verified by the third party organization like international standard organizations.

Software quality assurance focuses on:

- software's portability
- software's usability
- software's reusability
- software's correctness
- software's maintainability
- software's error control

Software Quality Assurance has:

1. A quality management approach
2. Formal technical reviews
3. Multi testing strategy
4. Effective software engineering technology
5. Measurement and reporting mechanism

Major Software Quality Assurance Activities:

1. **SQA Management Plan:**
   Make a plan for how you will carry out the sqa through out the project.

Think about which set of software engineering activities are the best for project. check level of sqa team skills.

2. **Set The Check Points:**
   SQA team should set checkpoints. Evaluate the performance of the project on the basis of collected data on different check points.

3. **Multi testing Strategy:**
   Do not depend on a single testing approach. When you have a lot of testing approaches available use them.

4. **Measure Change Impact:**
   The changes for making the correction of an error sometimes re introduces more errors keep the measure of impact of change on project. Reset the new change to change check the compatibility of this fix with whole project.

5. **Manage Good Relations:**
   In the working environment managing good relations with other teams involved in the project development is mandatory. Bad relation of sqa team with programmers team will impact directly and badly on project. Don't play politics.

Benefits of Software Quality Assurance (SQA):

1. SQA produces high quality software.
2. High quality application saves time and cost.
3. SQA is beneficial for better reliability.
4. SQA is beneficial in the condition of no maintenance for a long time.
5. High quality commercial software increase market share of company.
6. Improving the process of creating software.
7. Improves the quality of the software.

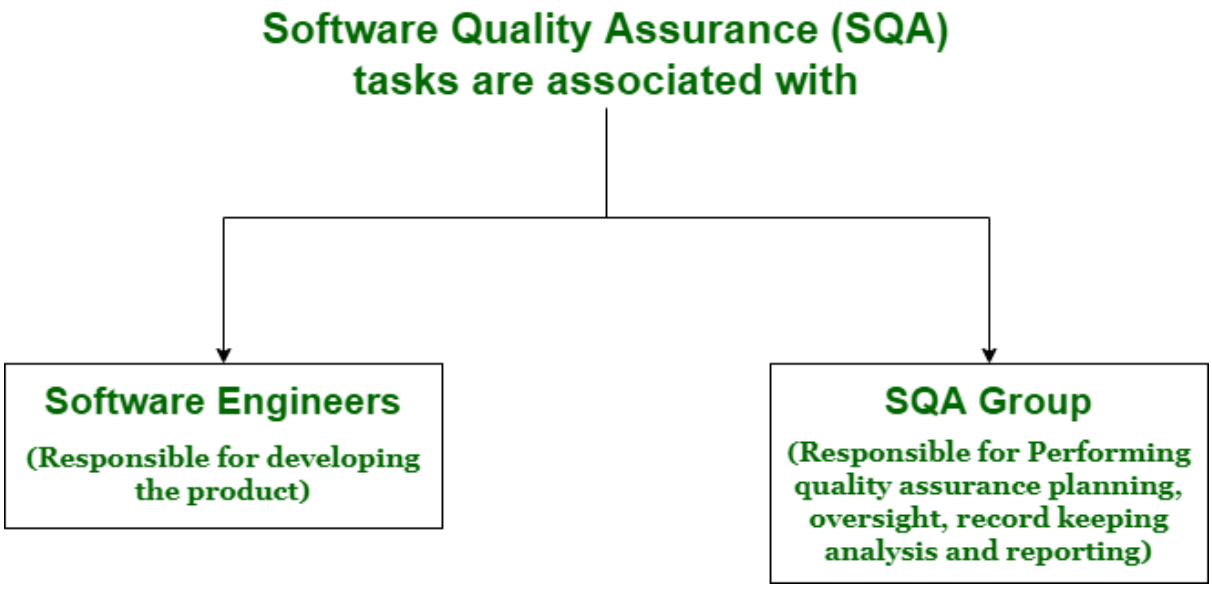
8. It cuts maintenance costs. Get the release right the first time, and your company can forget about it and move on to the next big thing. Release a product with chronic issues, and your business bogs down in a costly, time-consuming, never-ending cycle of repairs.

Disadvantage of SQA:
There are a number of disadvantages of quality assurance. Some of them include adding more resources, employing more workers to help maintain quality and so much more.

Goals and Measures of Software Quality Assurance

Software Quality simply means to measure how well software is designed i.e. the quality of design, and how well software conforms to that design i.e. quality of conformance. Software quality describes degree at which component of software meets specified requirement and user or customers' needs and expectations.

Software Quality Assurance (SQA) is a planned and systematic pattern of activities that are necessary to provide a high degree of confidence regarding quality of a product. It actually provides or gives a quality assessment of quality control activities and helps in determining validity of data or procedures for determining quality. It generally monitors software processes and methods that are used in a project to ensure or assure and maintain quality of software.



Goals of Software Quality Assurance :

- Quality assurance consists of a set of reporting and auditing functions.
- These functions are useful for assessing and controlling effectiveness and completeness of quality control activities.
- It ensures management of data which is important for product quality.
- It also ensures that software which is developed, does it meet and compiles with standard quality assurance.
- It ensures that end result or product meets and satisfies user and business requirements.
- It simply finds or identify defects or bugs, and reduces effect of these defects.

Measures of Software Quality Assurance :

There are various measures of software quality. These are given below:

1. **Reliability –**
   It includes aspects such as availability, accuracy, and recoverability of system to continue functioning under specific use over a given period of time. For example, recoverability of system from shut-down failure is a reliability measure.
2. **Performance –**
   It means to measure throughput of system using system response time,

recovery time, and start up time. It is a type of testing done to measure performance of system under a heavy workload in terms of responsiveness and stability.

3. **Functionality –**
   It represents that system is satisfying main functional requirements. It simply refers to required and specified capabilities of a system.
4. **Supportability –**
   There are a number of other requirements or attributes that software system must satisfy. These include- testability, adaptability, maintainability, scalability, and so on. These requirements generally enhance capability to support software.
5. **Usability –**
   It is capability or degree to which a software system is easy to understand and used by its specified users or customers to achieve specified goals with effectiveness, efficiency, and satisfaction. It includes aesthetics, consistency, documentation, and responsiveness.

Software Technical Reviews in Software Testing

Software Technical Review :

A software technical review is examined by a team of qualified software engineers for the suitability of the software product. This process can also be defined as a critical evaluation of an object in the software. Through the software technical review process, we can identify the errors or defects in the software product in the early phase itself.

Types of STRs :

The intent of this section is for the students to realize that the types of reviews that must be performed on a project are dependent upon the specific intermediate deliverables that are produced. Various application areas should also be described in the context of their review processes. The types of developmental modes are as follows:

1. **Development life cycle models –**
   a. Waterfall model
   b. Rapid prototyping
   c. Iterative enhancement
   d. Maintenance activity modeling
2. **Current Standards –**
   a. Military standards
   b. IEEE standards
   c. NBS standards

Furthermore, the reviews are classified into two types: Formal and Informal reviews.
Informal reviews are meant to describe the type of review that typically occurs spontaneously among peers and in which the reviewers have any work and also not create a review report. **Formal reviews** are characterized by carefully planned meetings in which reviewers are held accountable for their participation in the review and in which a review report containing action items is generated and acted upon.

Informal reviews	Formal reviews
It is a type of review that typically occurs spontaneously among teams.	It is a type of review that is done by a team of software testers or the reviewer's team.
It is generally done by Software developers or engineers.	It is done by the Reviewers and they are responsible for this review.
Here, no report is created as it is informal.	Here, a report is created as it is formal.

Importance of STRs :

Some of the reasons behind the importance of STRs are as follows:

1. It also impacts employee morale. For some employees, such as maintenance personnel, the reviews may provide an opportunity to gain visibility of their work and, thus, will be viewed positively.
2. It helps in software maintainability as it improves the developer's general understanding of the whole system, which further facilitates error diagnosis during maintenance.
3. It helps in the tracking of a project for both project management and customers. With this, we can track management, customers, and also developers' projects.
4. They provide feedback about the software and its development process. Examples of how review processes can impact the existing software development such as by identifying weaknesses in the software that will require additional validation effort in the future must also be provided.

Review Methodologies :

There are many variations to performing technical reviews. Most of these approaches involve a group meeting to assess a work product; however, variations of reviews exist that do not require a review group

meeting. The three major approaches for performing the software technical review are given as follows:

1. **Walk-through –**
   Walk-throughs are a formal and very systematic type of verification method as compared to peer review. In a walkthrough, the author of the software document presents the document to other persons which can range from 2 to 7.
2. **Inspections –**
   Inspections are the most structured and formal type of verification method and are commonly known as inspections. It requires detailed preparation for the reviewing team members and it includes a very high systematic review of the software product.
3. **Audits –**
   It can also be described as an external type of review process as it serves to ensure that the software is correctly verified and working as per the requirement.

Software Testing – Bug vs Defect vs Error vs Fault vs Failure

Software Testing defines a set of procedures and methods that check whether the actual software product matches with expected requirements, thereby ensuring that the product is Defect free. There are a set of procedures that needs to be in mind while testing the software manually or by using automated procedures. The main purpose of software testing is to identify errors, deficiencies, or missing requirements with respect to actual requirements. Software Testing is Important because if there are any bugs or errors in the software, they can be identified early and can be solved before the delivery of the software product. The article focuses on discussing the difference between bug, defect, error, fault, and failure.

What is a Bug?

A bug refers to defects which means that the software product or the application is not working as per the adhered requirements set. When we have any type of logical error, it causes our code to break, which results in a bug. It is now that the Automation/ Manual Test Engineers describe this situation as a bug.

- A bug once detected can be reproduced with the help of standard bug-reporting templates.
- Major bugs are treated as prioritized and urgent especially when there is a risk of user dissatisfaction.
- The most common type of bug is a crash.
- Typos are also bugs that seem tiny but are capable of creating disastrous results.

What is a Defect?

A defect refers to a situation when the application is not working as per the requirement and the actual and expected result of the application or software are not in sync with each other.

- The defect is an issue in application coding that can affect the whole program.
- It represents the efficiency and inability of the application to meet the criteria and prevent the software from performing the desired work.
- The defect can arise when a developer makes major or minor mistakes during the development phase.

What is an Error?

Error is a situation that happens when the Development team or the developer fails to understand a requirement definition and hence that misunderstanding gets translated into buggy code. This situation is referred to as an Error and is mainly a term coined by the developers.

- Errors are generated due to wrong logic, syntax, or loop that can impact the end-user experience.
- It is calculated by differentiating between the expected results and the actual results.

- It raises due to several reasons like design issues, coding issues, or system specification issues and leads to issues in the application.

  What is a Fault?

  Sometimes due to certain factors such as Lack of resources or not following proper steps Fault occurs in software which means that the logic was not incorporated to handle the errors in the application. This is an undesirable situation, but it mainly happens due to invalid documented steps or a lack of data definitions.

- It is an unintended behavior by an application program.
- It causes a warning in the program.
- If a fault is left untreated it may lead to failure in the working of the deployed code.
- A minor fault in some cases may lead to high-end error.
- There are several ways to prevent faults like adopting programming techniques, development methodologies, peer review, and code analysis.

  What is a Failure?

  Failure is the accumulation of several defects that ultimately lead to Software failure and results in the loss of information in critical modules thereby making the system unresponsive. Generally, such situations happen very rarely because before releasing a product all possible scenarios and test cases for the code are simulated. Failure is detected by end-users once they face a particular issue in the software.

- Failure can happen due to human errors or can also be caused intentionally in the system by an individual.
- It is a term that comes after the production stage of the software.
- It can be identified in the application when the defective part is executed.

  A simple diagram depicting Bug vs Defect vs Fault vs Failure:

Bug vs Defect vs Error vs Fault vs Failure

Some of the vital differences between bug, defect, fault, error, and failure are listed in the below table:

Basis	Bug	Defect	Fault	Error	Failure
Definition	A bug refers to defects which means that the software product or the application is not working as per the adhered requirements set	A Defect is a deviation between the actual and expected output	A Fault is a state that causes the software to fail and therefore it does not achieve its necessary function.	An Error is a mistake made in the code due to which compilation or execution fails,	Failure is the accumulation of several defects that ultimately lead to Software failure and results in the loss of information in critical modules thereby making the system unresponsive.
Raised by	Test Engineers	The defect is identified by The Testers And is resolved by developers in the development phase of SDLC.	Human mistakes lead to fault.	Developers and automation test engineers	The failure is found by the test engineer during the development cycle of SDLC
Different types	Logical bugs Algorithmic bugs Resource bugs	Defects are classified as follows: Based on Priority: High Medium	Business Logic Faults Functional and Logical Faults Graphical User	Syntactic Error UI screen error Error handling error Flow control error	NA

Basis	Bug	Defect	Fault	Error	Failure
		Low Based on Severity: Critical Major Minor Trivial	Interface (GUI) Faults Performance Faults Security Faults Hardware Faults	Calculation error Hardware error	
Reasons behind	Missing Logic Erroneous Logic Redundant codes	Receiving & providing incorrect input Coding/Logical Error leading to the breakdown of software	Wrong design of the data definition processes. An irregularity in Logic or gaps in the software leads to the non-functioning of the software.	Error in code. Inability to compile/execute a program Ambiguity in code logic Misunderstanding of requirements Faulty design and architecture Logical error	Environment variables System Errors Human Error
Way to prevent the reasons	Implementing Test-driven development. Adjusting enhanced development practices and evaluation of cleanliness of the code.	Implementing Out-of-the-box programming methods. Proper usage of primary and correct software coding practices.	Peer review of the Test documents and requirements. Verifying the correctness of software design and coding.	Conduct peer reviews and code-reviews Need for validation of bug fixes and enhancing the overall quality of the software.	Confirmation of Re-testing the process end to end, Carefully review the requirements as well as the specifications. Categorizing and evaluating the errors and issues.