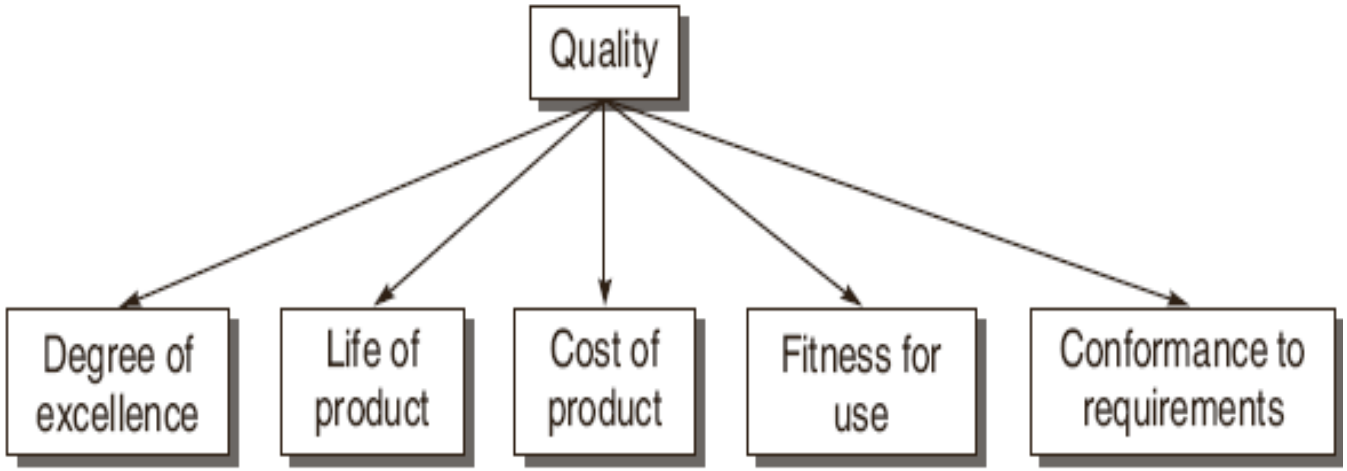
Software Quality Management

Module 5

Quality as Multidimensional Concept



The degree to which a product or service possesses a desired combination of attributes

What is Quality?

- "an inherent or distinguishing characteristic; a property; having a high degree of excellence"
- Features & functionality
  - "fitness for use"
  - "conformance to requirements"

Broadening the Concept of Quality

- Product Quality
- Process Quality

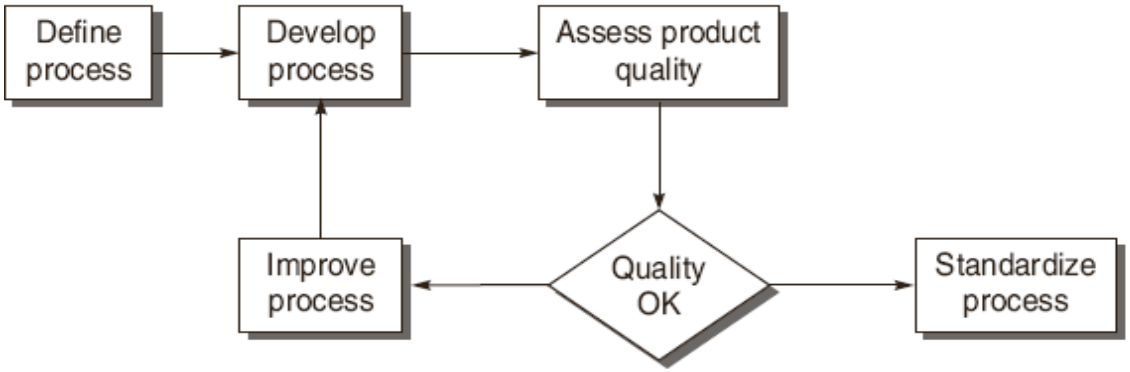


Figure 13.2 Process affects quality

Five Views of Software Quality

- Transcendental view
- User view
- Manufacturing view
- Product view
- Value-based view

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Five Views of Software Quality

- Transcendental view
  - Quality is something that can be recognized through experience, but not defined in some tractable form.
  - A good quality object stands out, and it is easily recognized.
  - It is "something toward which we strive as an ideal, but may never implement completely“
  - It's mainly feelings about something.
- User view
  - Quality concerns the extent to which a product meets user needs and expectations.
  - Is a product fit for use?
  - This view is highly personalized.
    - A product is of good quality if it satisfies a large number of users.
    - It is useful to identify the product attributes which the users consider to be important.
  - This view may encompass many subject elements, such as *usability*, *reliability*, and *efficiency*.

Transcendental view

It's like I can’t define it, but I know when I see it.. The examples of this point of view are: "I love this product", "I feel beautiful" (after usage of some cosmetic). It's mainly feelings about something.

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Five Views of Software Quality

- Manufacturing view
  - This view has its genesis in the manufacturing industry – auto and electronics.
  - Key idea: Does a product satisfy the requirements?
    - Any deviation from the requirements is seen as reducing the quality of the product.
  - The concept of process plays a key role.
  - Products are manufactured "right the first time" so that the cost is reduced
    - Development cost
    - Maintenance cost
  - Conformance to requirements leads to uniformity in products.
  - Product quality can be incrementally improved by improving the process.
    - The CMM and ISO 9001 models are based on the manufacturing view.

Five Views of Software Quality

- Product view
  - Hypothesis: If a product is manufactured with good internal properties, then it will have good external properties.
  - One can explore the causal relationship between *internal properties* and *external qualities*.
  - Example: *Modularity* enables *testability*.

- Value-based view
  - This represents the merger of two concepts: *excellence* and *worth*.
  - Quality is a measure of excellence, and value is a measure of worth.
  - Central idea
    - How much a customer is willing to pay for a certain level of quality.
    - Quality is meaningless if a product does not make economic sense.
    - The value-based view makes a trade-off between cost and quality.

McCall's Quality Factors and Criteria

- Quality Factors
  - McCall, Richards, and Walters studied the concept of software quality in terms of two key concepts as follows:
    - *quality factors*, and
    - *quality criteria*.
  - A quality factor represents the behavioral characteristic of a system.
    - Examples: correctness, reliability, efficiency, testability, portability, …

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

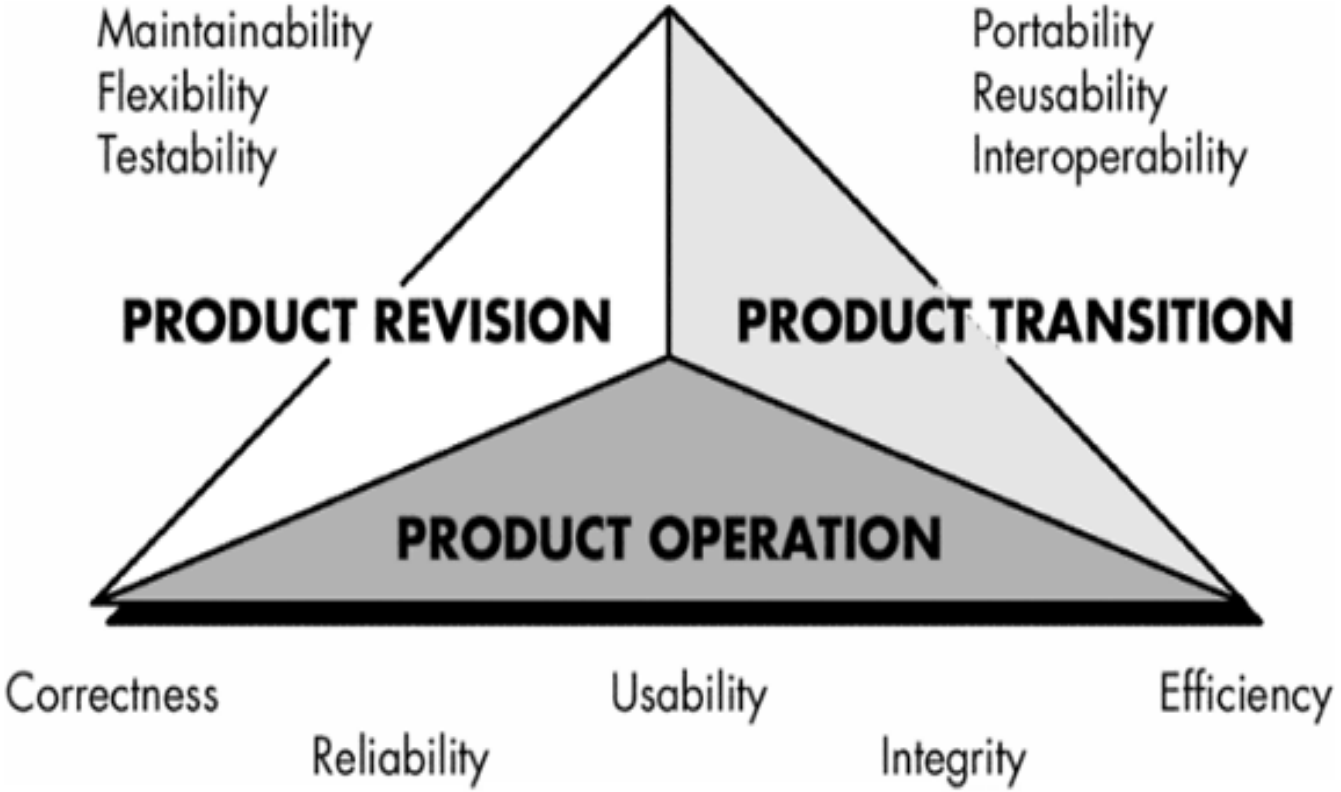
M

Quality Factors	Definitions
Correctness	The extent to which a program satisfies its specifications and fulfills the user's mission objectives.
Reliability	The extent to which a program can be expected to perform its intended function with required precision.
Efficiency	The amount of computing resources and code required by a program to perform a function.
Integrity	The extent to which access to software or data by unauthorized persons can be controlled.
Usability	The effort required to learn, operate, prepare input, and interpret output of a program.
Maintainability	The effort required to locate and fix a defect in an operational program.
Testability	The effort required to test a program to ensure that it performs its intended functions.
Flexibility	The effort required to modify an operational program.
Portability	The effort required to transfer a program from one hardware and/ or software environment to another.
Reusability	The extent to which parts of a software system can be reused in other applications.
Interoperability	The effort required to couple one system with another.

Table : McCall's quality factors

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

McCall's Quality Factors



Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

McCall's Quality Factors and Criteria

Quality Categories	Quality Factors	Broad Objectives
Product Operation	Correctness	Does it do what the customer wants?
	Reliability	Does it do it accurately all of the time?
	Efficiency	Does it quickly solve the intended problem?
	Integrity	Is it secure?
	Usability	Can I run it?
Product Revision	Maintainability	Can it be fixed?
	Testability	Can it be tested?
	Flexibility	Can it be changed?
Product Transition	Portability	Can it be used on another machine?
	Reusability	Can parts of it be reused?
	Interoperability	Can it interface with another system?

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

McCall's Quality Factors and Criteria

- Quality Criteria
  - A quality criterion is an attribute of a quality factor that is related to software development.
  - Example:
    - Modularity is an attribute of the architecture of a software system.
    - A highly modular software allows designers to put cohesive components in one module, thereby increasing the maintainability of the system.
    - In Table:- quality criteria have been listed.

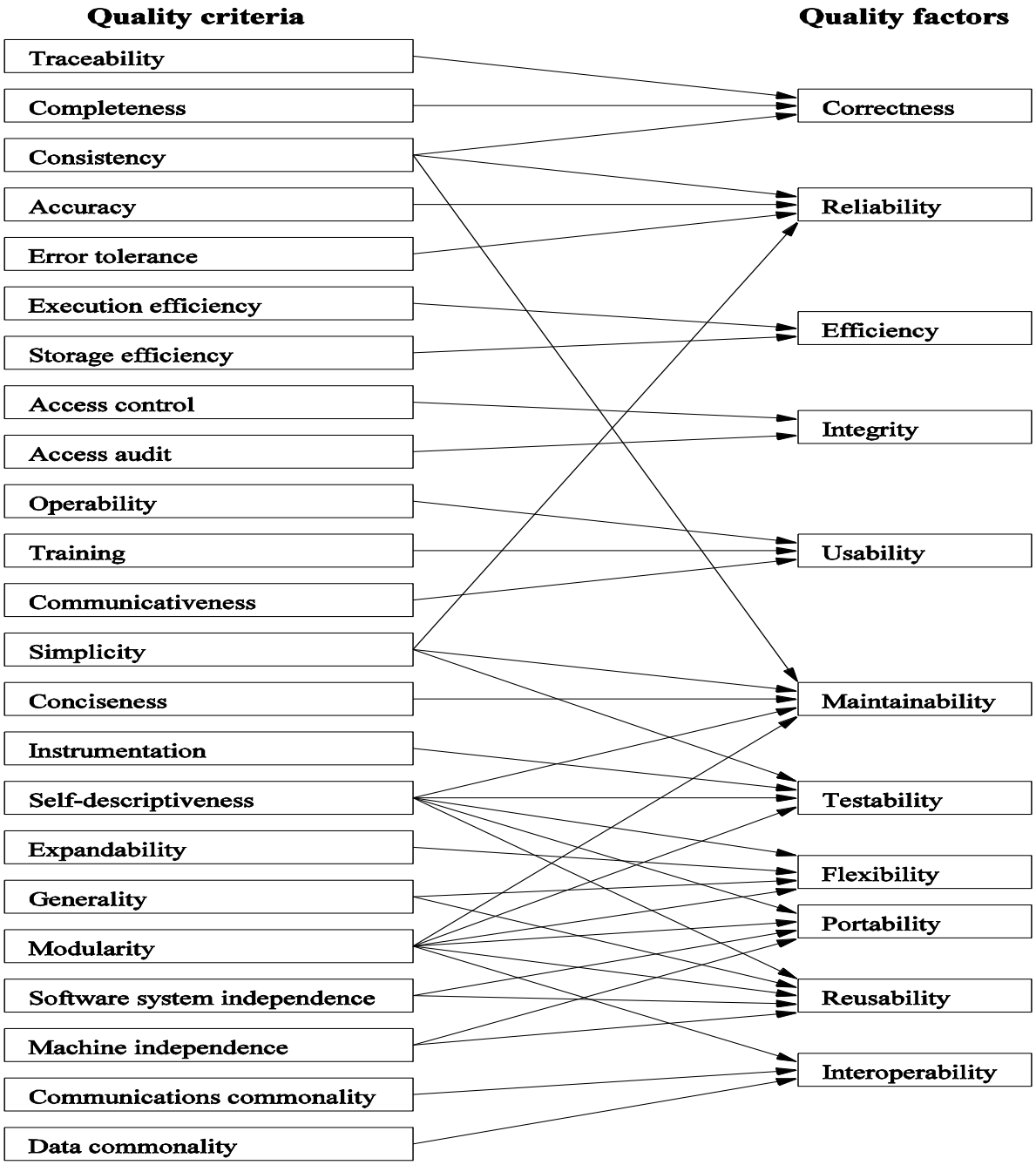
Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Quality Criteria	Definitions of Quality Criteria
Access audit	The ease with which software and data can be checked for compliance with standards or other requirements.
Access control	The provisions for control and protection of the software and data.
Accuracy	The precisions of computations and output.
Communication commonality	The degree to which standard protocols and interfaces are used.
Completeness	The degree to which a full implementation of the required functionalities has been achieved.
Communicativeness	The ease with which inputs and outputs can be assimilated.
Conciseness	The compactness of the source code, in terms of lines of code.
Consistency	The use of uniform design and implementation techniques and notations throughout a project.
Data commonality	The use of standard data representations.
Error tolerance	The degree to which continuity of operation is ensured under adverse conditions.
Execution efficiency	The run-time efficiency of the software.
Expandability	The degree to which storage requirements or software functions can be expanded.
Generality	The breadth of the potential application of software components.
Hardware independence	The degree to which the software is dependent on the underlying hardware.
Instrumentation	The degree to which the software provides for measurements of its use or identification of errors.
Modularity	The provision of highly independent modules.
Operability	The ease of operation of the software.
Self-documentation	The provision of in-line documentation that explains the implementation of components.
Simplicity	The ease with which the software can be understood.
Software system independence	The degree to which the software is independent of its software environment – non-standard language constructs, operating system, libraries, database management system, etc.
Software efficiency	The run-time storage requirements of the software.
Traceability	The ability to link software components to requirements.
Training	The ease with which new users can use the system.

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

McCall's Quality Factors and Criteria

- Relationship Between Quality Factors and Quality Criteria
  - Each quality factor is positively influenced by a set of quality criteria, and the same quality criterion impacts a number of quality factors.
    - Example: Simplicity impacts reliability, usability, and testability.
  - If an effort is made to improve one quality factor, another quality factor may be degraded.
    - Portable code may be less efficient.
  - Some quality factors positively impact others.
    - An effort to improve the correctness of a system will increase its reliability.



McCall's Quality Factors and Criteria

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University

Pooja Malhotra

Quality Management

Quality management is to

•plan suitable Quality control and Quality assurance activities,

•define procedures and standards which should be used during software development and verifying that these are being followed by everyone

•Properly execute and control activities.
•Verifying and improving the software process.

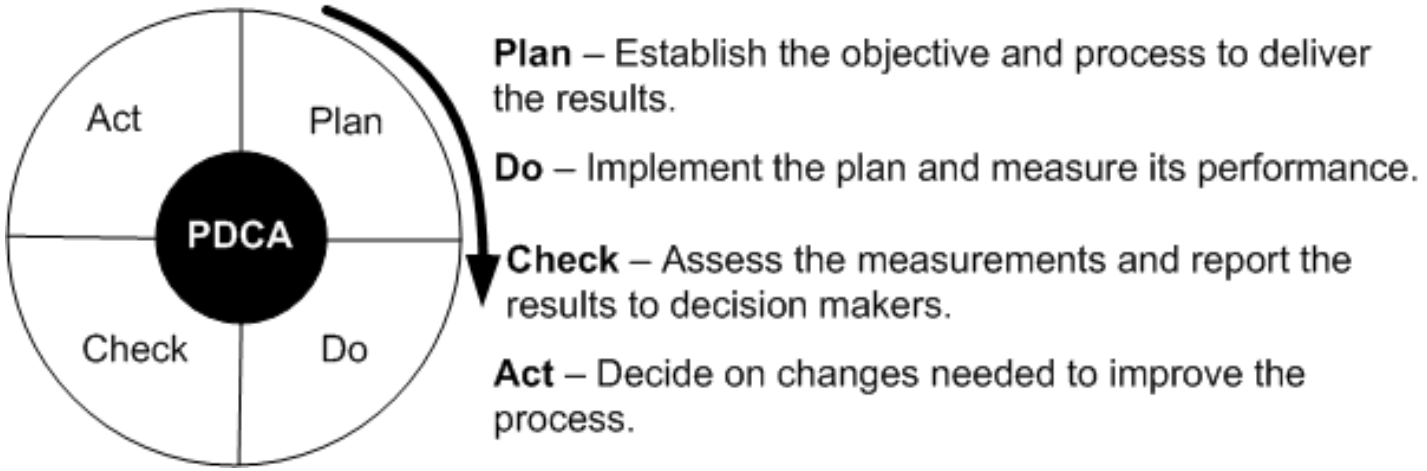
- Quality Management and Project Management

QM is different from project management (PM), in the sense that QM is not associated with a particular project, rather it is an organization-wide activity for many projects. On the other hand, PM is specific to a particular project and manages the cost schedule, resources, and quality of that project only.

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

The Quality Revolution - PDCA cycle

The Shewhart cycle



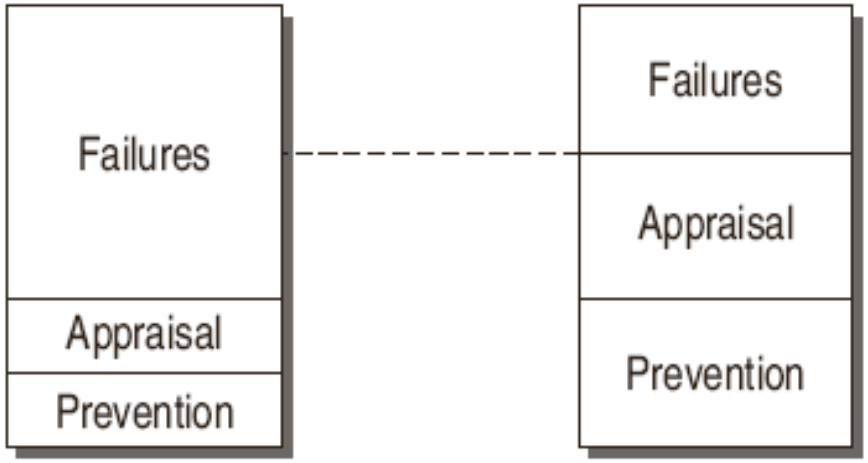
- Deming introduced Shewhart's PDCA cycle to Japanese researchers
- It illustrate the activity sequence:
  - Setting goals
  - Assigning them to measurable milestones
  - Assessing the progress against the milestones
  - Take action to improve the process in the next cycle

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Quality Cost & Benefits of investment on quality

Cost of quality is decomposed into three major categories:

1. Prevention Costs(Quality Planning, FTRs, Testing , training etc.)
2. Appraisal Costs(SMP, testing, inspection, equipment calibrations and maintenance)
3. Failure Costs(analyse and remove failures)
   - Internal Failure Costs(developer's site)
   - External Failure Costs(customer's site)



Benefits of investment on quality

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Quality Control and Quality Assurance

Pankaj Jalote [3], quality control focuses on finding and removing defects, whereas the main purpose of quality assurance is to verify that applicable procedures and standards are being followed. According to Pressman [7], a key concept of quality control is that all work products have defined, measurable specifications to which we may compare the output of each process. Quality assurance consists of auditing and reporting functions of management.

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Quality Control and Quality Assurance

Quality Control is basically related to software product such that there is minimum variation, according to the desired specifications. This variation is checked at each step of development. Quality control may include the following activities: Reviews, Testing using manual techniques or with automated tools (V & V).

Quality Assurance is largely related to the process. In addition, quality assurance activities are in management zone. Therefore auditing and reporting of quality based on quantitative measurements are also performed.

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Quality Control and Quality Assurance

The goal is to inform the management about quality-related issues and supply relevant data. It is the management's responsibility to address these problems. Thus, software quality assurance (S QA) is defined as a planned and systematic approach to evaluate quality and maintain software product standards, processes, and procedures.

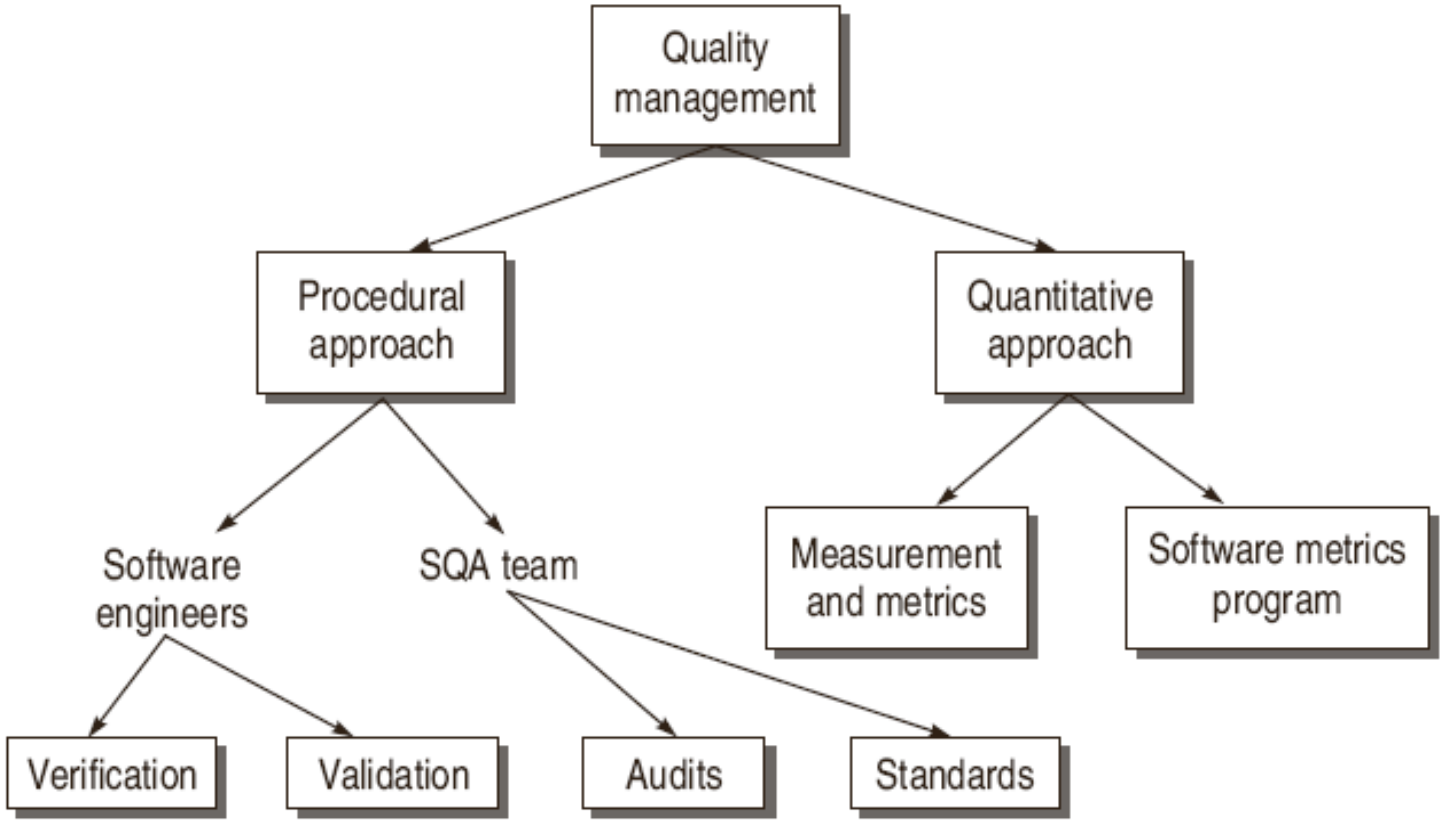
SQA includes the process of assuring that standards and procedures are established and are followed throughout the software acquisition life cycle.

Compliance with agreed-upon standards and procedures is evaluated through process monitoring, product evaluation, and audits.

Software development and control processes should include quality assurance approval points, where an SQA evaluation of the product may be done in relation to the applicable standards.

Quality assurance may include the following activities: Measuring the products and processes. Auditing the quality of every step in SDLC and the end-product. Verifying that the adopted procedures and standards are being followed.

Methods of quality management



Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Procedural Approach to QM

- SQA activities(Product evaluation and process monitoring)
  - Products- Standards and Process- Procedures
- SQA relationships with other assurance activities
  - Configuration Management monitoring
  - Verification and validation monitoring
  - Formal test monitoring
  - SQA during SDLC

Software quality assurance best practice

- **Continuous improvement:** All the standard process in SQA must be improved **frequently** and made **official** so that the other can follow. This process should be **certified** by popular organization such as ISO, CMMI… etc.
- **Documentation:** All the QA policies and methods, which are defined by QA team, should be documented for training and reuse for future projects.
- **Experience:** Choosing the members who are seasoned SQA auditors is a good way to ensure the quality of management review
- **Tool Usage:** Utilizing tool such as the tracking tool, management tool for SQA process reduces SQA effort and project cost.
- **Metrics:** Developing and creating metrics to track the software quality in its current state, as well as to compare the improvement with previous versions, will help increase the value and maturity of the Testing process
- **Responsibility:** The SQA process is not the SQA member's task, but **everyone**'s task. Everybody in the team is responsible for quality of product, not just the test lead or manager.

Quantitative Approach to QM

Major Issues

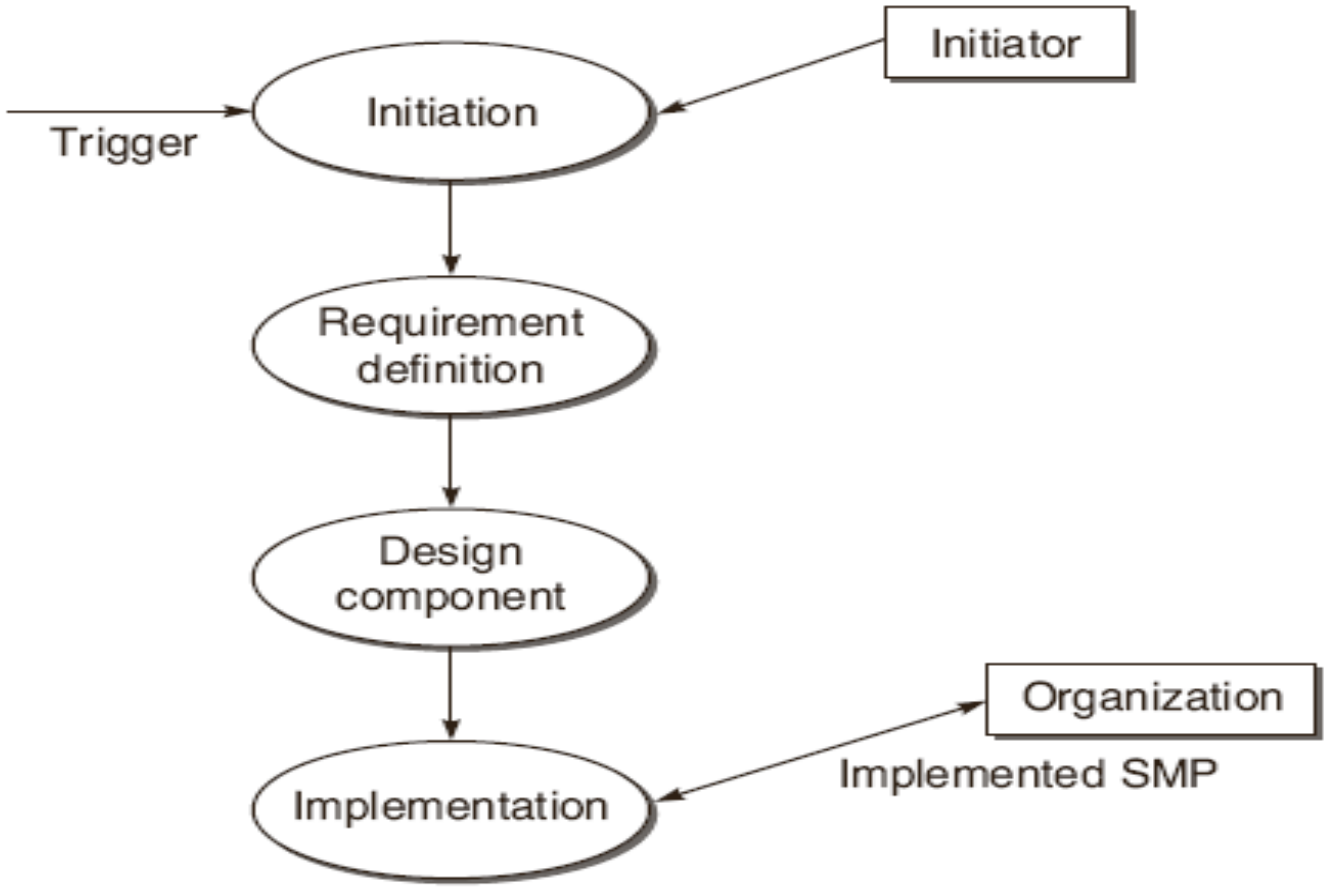
- Setting Quality Goal

Estimate for defects(P)current Project=

Defects(SP)X effort estimate(P)/Actual effort(SP)

- Managing software development process quantitatively- Intermediate goals

Paul Goodman Model for Software Metrics Program



Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Software Quality Metrics

Product Quality Metrics

- **Mean time to failure (MTTF)**

  MTTF metric is an estimate of the average or mean time until a product's first failure occurs.

Defect Density Metrics

- Defect Density = Number of Defects / Size of Product

Customer problem metrics

- problems per user month (PUM) = Total problems reported by the customer for a time period / Total number of licensed months of the software during the period.
- **Customer Satisfaction Metrics**

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Software Quality Metrics

In-Process Quality Metrics

- **Defect Density during testing**
- **Defect Arrival pattern during Testing**
- **Defect Removal Efficiency**

  DRE = (Defects removed during the month / No. of problems arrivals during the month) X 100

Software Quality Metrics

Metrics for Software maintenance

Fix backlog and backlog management index

BMI = (Number of problems closed during the month / Number of problem arrivals during the month) x 100 %

Fix response time and fix responsiveness

Percent Delinquent fixes

Percent Delinquent fixes = (Number of fixes that exceeded the response time criteria by severity level / Number of fixes delivered in a specified time) X 100%

Fix Quality: Defective fix percentage

6-Sigma

- Originated by Motorola in Schaumburg, IL
- Based on competitive pressures in 1980s – "Our quality stinks"

Sigma	Defects Per Million
1 δ	690,000
2 δ	308,537
3 δ	66,807
4 δ	6,210
5 δ	233
6 δ	3.4

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

6-Sigma

3 δ	6 δ
Five short or long landings at any major airport	One short or long landing in 10 years at all airports in the US
Approximately 1,350 poorly performed surgical operations in one week	One incorrect surgical operation in 20 years
Over 40,500 newborn babies dropped by doctors or nurses each year	Three newborn babies dropped by doctors or nurses in 100 years
Drinking water unsafe to drink for about 2 hours each month	Water unsafe to drink for one second every six years

6-Sigma D-M-A-I-C Cycle

- Define
  - The first step is to define customer satisfaction goals and subgoals; for example, reduce cycle time, costs, or defects. These goals then provide a baseline or benchmark for the process improvement.
- Measure
  - The Six Sigma team is responsible for identifying a set of *relevant* metrics.
- Analyze
  - With data in hand, the team can analyze the data for trends, patterns, or relationships. Statistical analysis allows for testing hypotheses, modeling, or conducting experiments.
- Improve
  - Based on solid evidence, improvements can be proposed and implemented. The Measure-Analyze-Improve steps are generally iterative to achieve target levels of performance.
- Control
  - Once target levels of performance are achieved, control methods and tools are put into place in order to maintain performance.

6-Sigma Roles & Responsibilities

- Master black belts
  - People within the organization who have the highest level of technical and organizational experience and expertise. Master black belts train black.
- Black belts
  - Should be technically competent and held in high esteem by their peers. They are actively involved in the Six Sigma change process.
- Green belts
  - Are Six Sigma team leaders or project managers. Black belts generally help green belts choose their projects, attend training with them, and then assist them with their projects once the project begins.
- Champions
  - Leaders who are committed to the success of the Six Sigma project and can ensure that barriers to the Six Sigma project are removed. Usually a high-level manager who can remove obstacles that may involve funding, support, bureaucracy, or other issues that black belts are unable to solve on their own.

Software Total Quality Management (STQM)

TQM is defined as a quality-centered, customer-focused, fact-based, team-driven, senior-management-led process to achieve an organization's strategic imperative through continuous process improvement.

T = Total = everyone in the organization

Q = Quality = customer satisfaction

M = Management = people and processes

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Software Total Quality Management (STQM)

The elements of TQM / STQM

- Customer-focus/Customer-focus in software development
- Process / Process, technology, and development quality
- Human-side of quality/Human-side of software quality
- Measurement and analysis

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

TEST MATURITY MODEL (TMM)

NEED FOR TEST PROCESS MATURITY

The basic underlying goal for every process is to have continuous process improvement. The testing process should also be designed such that there is scope of its continuous improvement.

The following are some of the reasons why test process maturity models are important:

- The progress and quality of the whole test process should be measured and the results should be used as input for further test process improvement.
- The testing process steps, i.e. test planning, design, execution, etc. are in place. There is a systematic fl ow of these activities in the organization.
- Testing maturity model provides important information to the testers and management, enabling them to improve the software quality.
- The model should be able to improve the process such that it provides information about the bug's source, i.e. the exact location of their existence, leading to minimization of correction costs.
- The model should be able to improve the process such that it detects high-priority or criticality bugs early.

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

TEST MATURITY MODEL (TMM)

MEASUREMENT AND IMPROVEMENT OF A TEST PROCESS

Determine target focus areas of test process: Why do you want to measure or improve your current testing process? This question should be asked to realize what your long-term goal is. Without realizing the goals, you cannot improve the test process in reality. For example, you may have the target focus on shortening the time-frame of the test process. It may be possible that testing activities are not performed in order, so you want to have a proper test organization in place.

Analyse the testing practices in the current testing process: Look for the strong and weak points of the current t est process and analyse the aspects which are lacking.

Compare the current practices with a test maturity model: Every test activity should be compared with a test maturity model in order to get the best test practices in a process.

Determine key process areas: Based on the analysis done in the previous step, list the key process areas where the current process needs attention.

Develop a plan: Depending on the key process areas recognized, develop a plan and suggest changes to nullify all the weak points of a process. Based on the suggestions, a modified test process is developed.

Implement the plan: Implement the developed plan to improve the existing test process.

TEST MATURITY MODEL (TMM)

TMM was developed in 1996 at the Illinois Institute of Technology . It has two major components:

1. A maturity model in which five maturity levels are distinguished (as in CMM)
2. An assessment model containing an assessment procedure, an assessment instrument/questionnaire, team training, and selection criteria

TEST MATURITY MODEL (TMM)

TMM Components

The TMM structure contains several components, which are described here. Figure shows the framework of TMM.

Figure 14.4 Framework of TMM

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University

TEST MATURITY MODEL (TMM)

Maturity levels

Each maturity level represents a specific testing maturity in the evolution to a matured testing process. Each upper level assumes that the lower level has practices that have been executed.

There are five levels within TMM:

- Initial
- Phase definition
- Integration
- Management and measurement
- Optimization/defect prevention and quality control

TEST MATURITY MODEL (TMM)

Maturity goals

To achieve a certain maturity level in the model, some goals have been identified that must be addressed. The maturity goals of TMM are the same as what we call the key process areas in CMM. Goals are identified for each level, except for level 1.

Maturity sub-goals

For each maturity goal, the sub-goals that are more concrete are defined. They define the scope, boundaries, and accomplishments for a particular level.

Activities, tasks, and responsibilities (ATRs)

To implement the sub-goals identified, there should be a defined set of activities and responsibilities. Therefore, a set of ATRs are designed for each sub-goal. The ATRs address implementation and organizational adaptation issues at a specific level, targeting to improve the testing capability. The responsibility for these activities and tasks is assigned to the three key participants in the testing process: managers, developers/testers, and users/clients.

TEST MATURITY MODEL (TMM)

Examples of ATRs are:

- Upper management provides adequate resources and funding from the committee for testing and debugging (manager view).
- Developers/testers work with managers to set testing and debugging goals (developers/testers view).
- The testing goals for each level of testing: unit, integration, system, and acceptance are set (developers/testers view).
- Users/clients meet with developers/testers to give their view of the nature of testing/debugging and policies. These goals and policies should represent their interests with respect to product quality (users/clients view).

TEST MATURITY MODEL (TMM)

TMM Levels

TMM has been deliberately designed similar to CMM. The idea is to ensure the growth in testing maturity is in tune with the growth in software capability maturity. The characteristics of five TMM levels are given below, as shown in Fig.

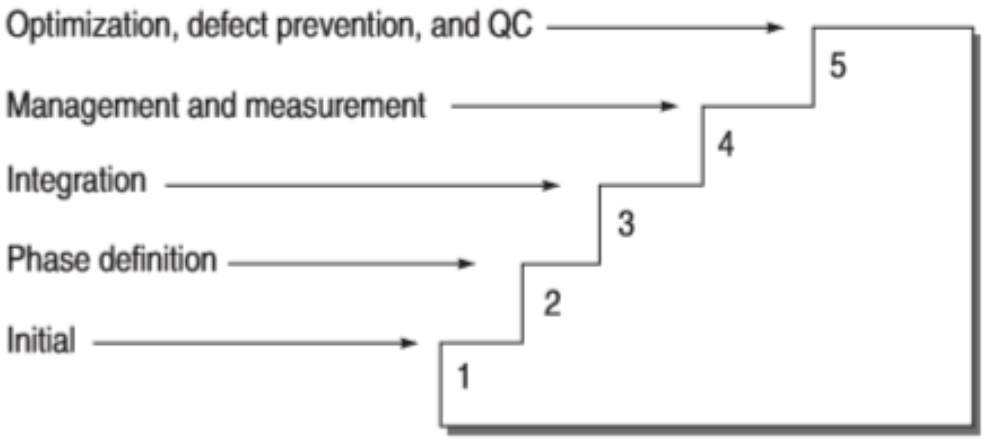


Figure 14.5 Five levels of TMM

TEST MATURITY MODEL (TMM)

1.Initial level

It is characterized by a random testing process. Testing activities are performed after coding in an ad-hoc way. The objective of testing at this level is to show only the functioning of the software.

2.Phase definition

Testing and debugging are considered separate activities at this level. Testing still follows coding, but is a planned activity. The primary goal of testing at this maturity level is to show that the software meets its specifications.

3.Integration

It assumes that testing is no longer a phase after coding; it is integrated into the entire software lifecycle and gets early testing consideration. Test objectives are established with respect to the requirements based on user and client needs and are used for test case plan, design, and success criteria. There is a test organization in place consisting of various persons with assigned responsibilities.

4.Management and measurement

At this level, testing is recognized as a measured and quantified process. Testing starts as soon as the SRS is prepared, i.e. reviews at all phases of the development process are now recognized as testing and quality control activities. Testware is maintained for reuse, defects are adequately logged.

TEST MATURITY MODEL (TMM)

5. Optimization, defect-prevention, and quality control

At this level, the testing process is defined and managed. The test process can be measured for costs and other parameters and effectiveness can be monitored. The concept of bug prevention instead of bug-detection and quality control are practiced. There are mechanisms put in place to fine-tune and improve the testing process.

Maturity goals and sub-goals

As discussed above, TMM levels include various goals and under each goal, there are sub-goals. These goals and sub-goals are necessary in order to measure a test process maturity and correspondingly focus on those goals and sub-goals in which the process is deficient. If everything is defined in the model in the form of goals, sub-goals, and ATRs, then it becomes easy to measure the maturity of the process.

Reference: Software Testing Principles and Practices, Naresh Chauhan, Oxford University Press, Gaurav Gupta

TEST MATURITY MODEL (TMM)

Table 14.1 TMM maturity goals by level

Maturity Level	Goals
Level 1	--
Level 2	Develop and establish a test policy. Develop testing and debugging goals. Initiate a testing planning process. Institutionalize basic testing techniques and methods.
Level 3	Establish a software test organization. Establish a testing training program. Integrate testing into software life cycle. Control and monitor test process.
Level 4	Establish an organization-wide review program. Establish a test measurement program. Software quality evaluation.
Level 5	Application of process data for defect-prevention. Quality control. Test process optimization.

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Test Maturity Model (TMM)

Table 14.2 KPAs at Level 2

KPAs	Associated activities
Develop testing and debugging goals	Various goals, tasks, activities, and tools for each must be identified. Responsibilities for each must be assigned.
Initiate a testing planning process	Focus on initiating a planning process which involves stating objectives, analysing risks, outlining strategies, and developing test design specifications and test cases. Also addresses the allocation of recourses and responsibilities for testing on the unit, integration system, and acceptance levels.
Institutionalize basic testing techniques and methods	Emphasis on applying basic testing techniques and methods to improve test process capability across the organization.

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Test Maturity Model (TMM)

Table 14.3 KPAs at Level 3

KPAs	Associated activities
Establish a software test organization	Establishment of a software test organization to identify a group of people who are responsible for testing.
Establish a testing training program	Establishment of a testing training program to ensure a skilled staff is available to the testing group.
Integrate testing into software life cycle	Integration of testing sub-phases into software life cycle. Also a mechanism must be established that allows testers to work with developers, which facilitates testing activity integration.
Control and monitor test process	Several controlling and monitoring activities are performed which provide visibility and ensure that the testing process proceeds according to plan.

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Test Maturity Model (TMM)

Table 14.4 KPAs at Level 4

KPAs	Associated activities
Establish an organization-wide review program	Establishment of a review program to remove defects from software work products and to test work products early and effectively.
Establish a test measurement program	Organizational–wide test measurement policies and goals must be defined as well as action plans that apply measurement results to test process improvements must be developed and documented.
Software quality evaluation	Software quality evaluation involves defining measurable quality attributes and defining quality goals to evaluate software work products. Quality goals are tied to testing process adequacy.

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Test Maturity Model (TMM)

Table 14.5 KPAs at Level 5

KPAs	Associated activities
Application of process data for defect-prevention	A defect-prevention team must be established with management support and the defects injected or removed must be identified and recorded during each phase.
Quality control	Organizations use statistical sampling, measurement of confidence levels, trustworthiness, and reliability goals to drive the testing process.
Test process optimization	Testing process is subjected to continuous improvement across projects and across the organization. A mechanism must be in place to evaluate new tools and technologies that may improve the capability and maturity of the testing process.

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Software Quality Tools

- Ishikawa Diagram
- Check List
- Control Chart
- Pareto Chart

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Cause & Effect Diagrams

Kaoru Ishikawa (1915 - 1989)

- Studied under Deming
- Believed quality is a continuous process that relies on all levels of the organization

Cause and effect diagrams (Ishikawa Diagram) are used for understanding organizational or business problem causes.

Organizations face problems everyday and it is required to understand the causes of these problems in order to solve them effectively. Cause and effect diagrams exercise is usually a teamwork.

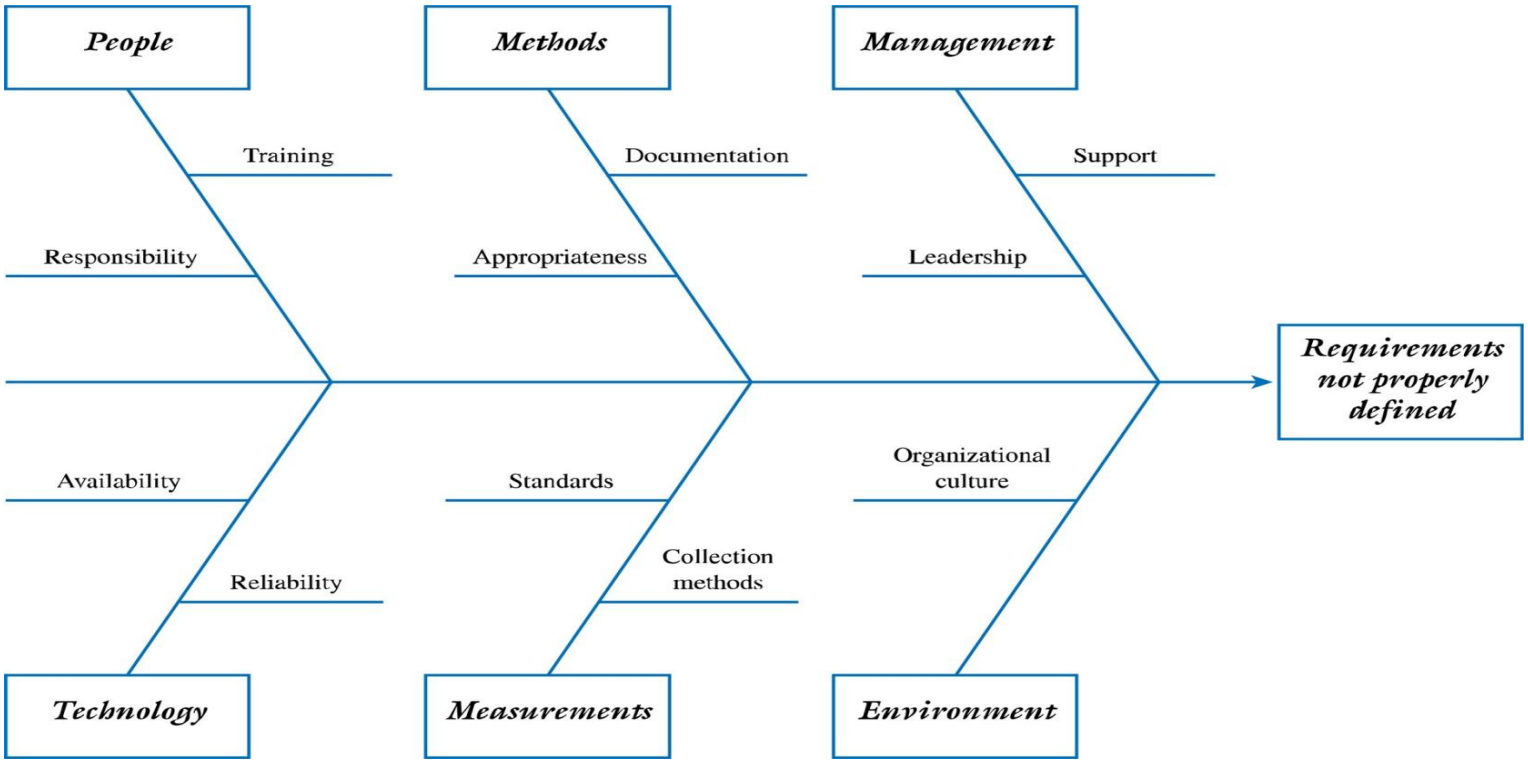
A brainstorming session is required in order to come up with an effective cause and effect diagram.

All the main components of a problem area are listed and possible causes from each area is listed.

Then, most likely causes of the problems are identified to carry out further analysis.

Ishikawa, or Fishbone Diagram

Best developed by brainstorming or by using a learning cycle approach



Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Control Charts

- Walter A. Shewhart (1891 – 1967)
  - Worked for Western Electric Company (Bell Telephones)
  - Introduced the concept of the control chart as a tool better to understand variation and to allow management to shift its focus away from inspection and more towards the prevention of problems and the improvement of processes.

Control Charts

A **quality control chart** is a graphic that depicts whether sampled products or processes are meeting their intended specifications and, if not, the degree by which they vary from those specifications.

The control chart is a graph used to study how a process changes over time. Data are plotted in time order. A control chart always has a central line for the average, an upper line for the upper control limit, and a lower line for the lower control limit. These lines are determined from historical data. By comparing current data to these lines, you can draw conclusions about whether the process variation is consistent (in control) or is unpredictable (out of control, affected by special causes of variation).

Control Charts

Common and special causes are the two distinct origins of variation in a process, as defined in the statistical thinking and methods of Walter A. Shewhart and W. Edwards Deming. Briefly, "common causes", also called Natural patterns, are the usual, historical, quantifiable variation in a system, while "special causes" are unusual, not previously observed, non-quantifiable variation.

Control Charts

Common causes

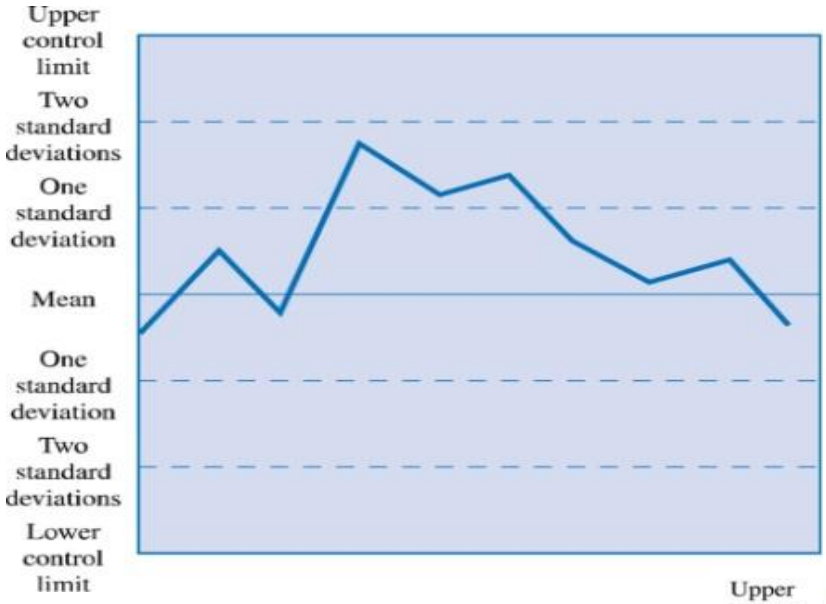
- Inappropriate procedures
- Poor design
- Poor maintenance of machines
- Measurement error
- Quality control error

Special causes

- Faulty controllers
- Machine malfunction
- Computer crash
- Poor batch of raw material

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Control Charts



Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Control Charts

Purpose:

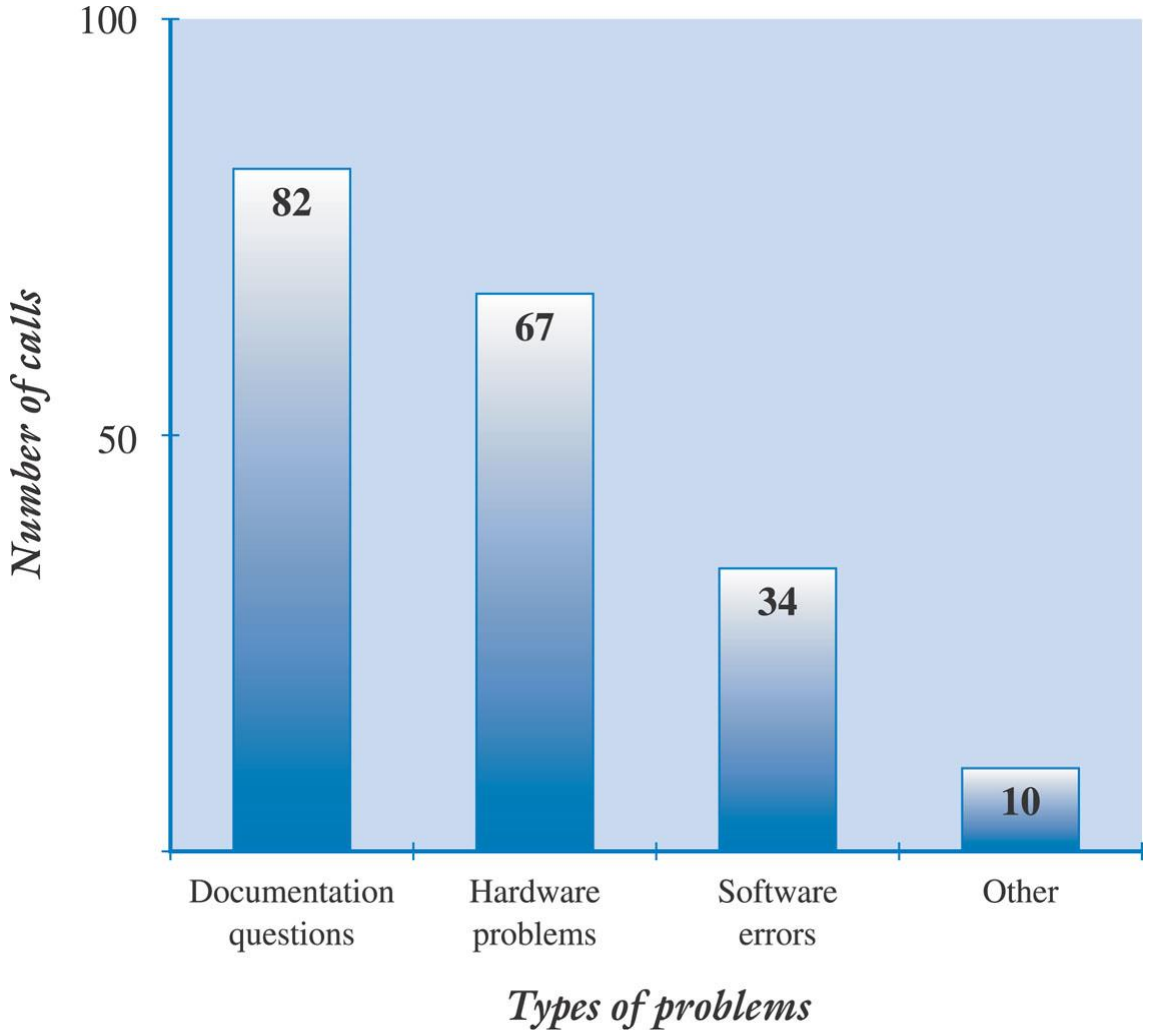
The primary purpose of a control chart is to predict expected product outcome.

Benefits:

- Predict process out of control and out of specification limits
- Distinguish between specific, identifiable causes of variation
- Can be used for statistical process control

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University Pooja Malhotra

Pareto Chart



Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

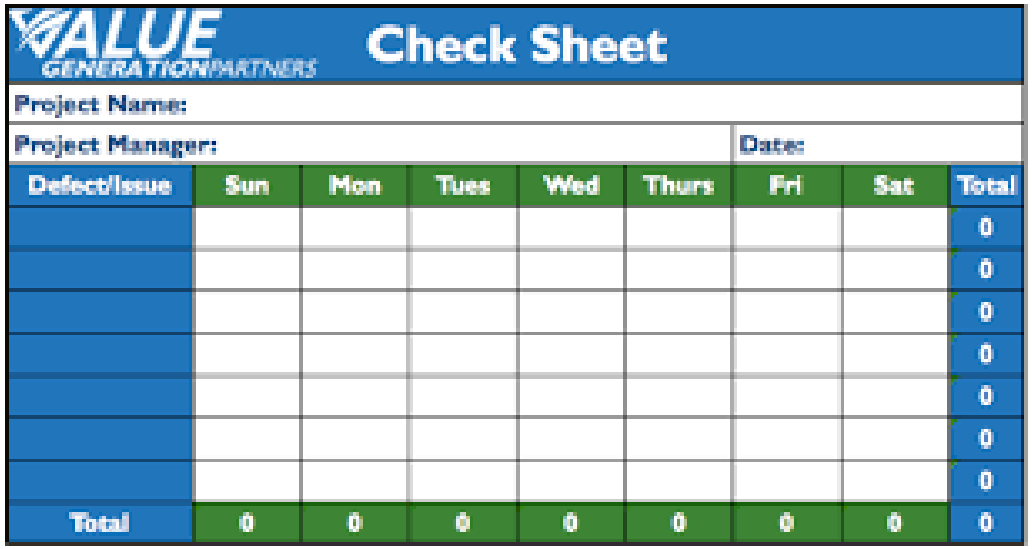
Pareto Chart

Pareto charts are used for identifying a set of priorities. You can chart any number of issues/variables related to a specific concern and record the number of occurrences.

This way you can figure out the parameters that have the highest impact on the specific concern.

This helps you to work on the propriety issues in order to get the condition under control.

Check List



VALUE GENERATION PARTNERS

Check Sheet

Project Name:							
Project Manager:						Date:	
Defect/Issue	Sun	Mon	Tues	Wed	Thurs	Fri	Sat	Total
								0
								0
								0
								0
								0
								0
								0
Total	0	0	0	0	0	0	0	0

Reference: Software Testing Principles and Practices, Naresh Chauhan , Oxford University
Pooja Malhotra

Check List/Sheet

A check sheet can be introduced as the most basic tool for quality. A check sheet is basically used for gathering and organizing data.

When this is done with the help of software packages such as Microsoft Excel, you can derive further analysis graphs and automate through macros available.

Therefore, it is always a good idea to use a software check sheet for information gathering and organizing needs.

One can always use a paper-based check sheet when the information gathered is only used for backup or storing purposes other than further processing.

References:

Software Testing Principles and Practices, Naresh Chauhan, Second edition, Oxford Higher Education