UNIT V

Metrics for Process and Products

Software Process and Product Metrics

What are metrics?

Software Process and Product Metrics are quantitative measures of-

- They are a management tool.
- They offer insight into the effectiveness of the software process and the projects that are conducted using the process as a framework.
- Basic quality and productivity data are collected.
- These data are analyzed, compared against past averages, and assessed.
- The goal is to determine whether quality and productivity improvements have occurred.
- The data can also be used to pinpoint problem areas.
- Remedies can then be developed and the software process can be improved.

Need for Software Metrics:

Following are the needs for the software Metrics.

To **characterize** in order to

- Gain an understanding of processes, products, resources, and environments.
- Establish baselines for comparisons with future assessments

To **evaluate** in order to

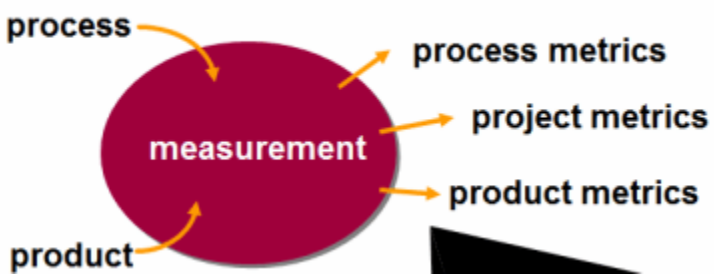
- Determine status with respect to plans

To **predict** in order to

- Gain understanding of relationships among processes and products.
- Build models of these relationships

To **improve** in order to

- Identify roadblocks, root causes, inefficiencies, and other opportunities for improving product quality and process performance.



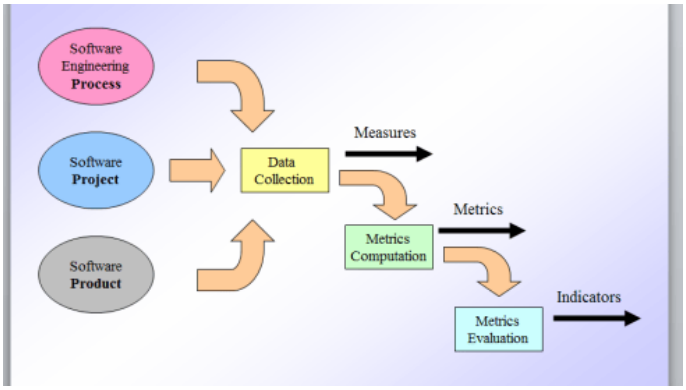
Metrics for Process:

Process metrics are collected across all projects and over long periods of time.

- They are used for making strategic decisions.
- The intent is to provide a set of process indicators that lead to long-term software process improvement.

The only way to know how/where to improve any process is to

- Measure specific attributes of the process.
- Develop a set of meaningful metrics based on these attributes.
- Use the metrics to provide indicators that will lead to a strategy for improvement.



The effectiveness of a Process:

We measure the effectiveness of a process by deriving a set of metrics based on outcomes of the process such as:

- Errors uncovered before release of the software.
- Defects delivered to and reported by the end users.
- Work products delivered.
- Human effort expended.
- Calendar time expended.
- Conformance to the schedule.
- Time and effort to complete each generic activity.

Product Metric:

They focus on the quality of deliverables.

Product metrics are combined across several projects to produce process metrics.

Metrics for the product:

- Measures of the Analysis Model.

- Complexity of the Design Model
- Code metrics.

Furthermore, Complexity of the Design Model is classified as-

- Internal algorithmic complexity.
- Architectural complexity.
- Data flow complexity.

Software Measurement and Metrics

Software Measurement: A measurement is a manifestation of the size, quantity, amount, or dimension of a particular attribute of a product or process.

Software measurement is a titrate impute of a characteristic of a software product or the software process.

It is an authority within software engineering.

The software measurement process is defined and governed by ISO Standard.

Software Measurement Principles:

The software measurement process can be characterized by five activities-

1) **Formulation:** The derivation of software measures and metrics appropriate for the representation of the software that is being considered.
2) **Collection:** The mechanism used to accumulate data required to derive the formulated metrics.
3) **Analysis:** The computation of metrics and the application of mathematical tools.
4) **Interpretation:** The evaluation of metrics resulting in insight into the quality of the representation.
5) **Feedback:** Recommendation derived from the interpretation of product metrics transmitted to the software team.

Need for Software Measurement:

Software is measured to:

- Create the quality of the current product or process.
- Anticipate future qualities of the product or process.
- Enhance the quality of a product or process.
- Regulate the state of the project in relation to budget and schedule.

Classification of Software Measurement:

There are 2 types of software measurement:

1. **Direct Measurement:** In direct measurement, the product, process, or thing is measured directly using a standard scale.
2. **Indirect Measurement:** In indirect measurement, the quantity or quality to be measured is measured using related parameters i.e. by use of reference.

Metrics:

A metric is a measurement of the level at which any impute belongs to a system product or process.

Software metrics will be useful only if they are characterized effectively and validated so that their worth is proven.

There are 4 functions related to software metrics:

1. Planning
2. Organizing
3. Controlling
4. Improving

Characteristics of software Metrics:

1. **Quantitative:** Metrics must possess quantitative nature. It means metrics can be expressed in values.
2. **Understandable:** Metric computation should be easily understood, and the method of computing metrics should be clearly defined.
3. **Applicability:** Metrics should be applicable in the initial phases of the development of the software.
4. **Repeatable:** The metric values should be the same when measured repeatedly and consistent in nature.
5. **Economical:** The computation of metrics should be economical.
6. **Language Independent:** Metrics should not depend on any programming language.

Classification of Software Metrics:

There are 3 types of software metrics:

1. **Product Metrics:** Product metrics are used to evaluate the state of the product, tracing risks and

Undercover prospective problem areas. The ability of the team to control quality is evaluated.

2. **Process Metrics:** Process metrics pay particular attention to enhancing the long-term process of the team or organization.
3. **Project Metrics:** The project matrix describes the project characteristic and execution process.

- Number of software developer
- Staffing patterns over the life cycle of software
- Cost and schedule
- Productivity

Advantages of Software Metrics:

1. Reduction in cost or budget.
2. It helps to identify the particular area for improvising.
3. It helps to increase the product quality.
4. Managing the workloads and teams.
5. Reduction in overall time to produce the product.
6. It helps to determine the complexity of the code and to test the code with resources.
7. It helps in providing effective planning, controlling and managing of the entire product.

Disadvantages of Software Metrics:

1. It is expensive and difficult to implement the metrics in some cases.
2. Performance of the entire team or an individual from the team can't be determined. Only the performance of the product is determined.
3. Sometimes the quality of the product is not met with the expectation.
4. It leads to measure the unwanted data which is wastage of time.
5. Measuring the incorrect data leads to make wrong decision making.

Software Quality Metrics:

- Software quality metrics are a subset of software metrics that focus on the quality aspects of the product, process, and project.
- These are more closely associated with process and product metrics than with project metrics.
- Software quality metrics can be further divided into three categories −

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

Product Quality Metrics

This metrics include the following −

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

Mean Time to Failure

It is the time between failures.

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

Defect Density

It measures the defects relative to the software size expressed as lines of code or function point, etc. i.e., it measures code quality per unit.

This metric is used in many commercial software systems.

Customer Problems

It measures the problems that customers encounter when using the product.

It contains the customer's perspective towards the problem space of the software, which includes the non-defect oriented problems together with the defect problems.

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

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

Where,

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

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

Customer Satisfaction

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

- Very satisfied
- Satisfied
- Neutral
- Dissatisfied
- Very dissatisfied

Satisfaction with the overall quality of the product and its specific dimensions is usually obtained through various methods of customer surveys.

Based on the five-point-scale data, several metrics with slight variations can be constructed and used, depending on the purpose of analysis.

For example −

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

In-process Quality Metrics

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

- Defect density during machine testing
- Defect arrival pattern during machine testing

- Phase-based defect removal pattern
- Defect removal effectiveness

Defect density during machine testing

Defect rate during formal machine testing (testing after code is integrated into the system library) is correlated with the defect rate in the field.

Higher defect rates found during testing is an indicator that the software has experienced higher error injection during its development process, unless the higher testing defect rate is due to an extraordinary testing effort.

This simple metric of defects per KLOC or function point is a good indicator of quality, while the software is still being tested.

It is especially useful to monitor subsequent releases of a product in the same development organization.

Defect arrival pattern during machine testing

The overall defect density during testing will provide only the summary of the defects.

The pattern of defect arrivals gives more information about different quality levels in the field.

It includes the following −

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

Phase-based defect removal pattern

This is an extension of the defect density metric during testing.

In addition to testing, it tracks the defects at all phases of the development cycle, including the design reviews, code inspections, and formal verifications before testing.

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

The pattern of phase-based defect removal reflects the overall defect removal ability of the development process.

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

Defect removal effectiveness

It can be defined as follows −

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

This metric can be calculated for the entire development process, for the front-end before code integration and for each phase.

It is called **early defect removal** when used for the front-end and **phase effectiveness** for specific phases.

The higher the value of the metric, the more effective the development process and the fewer the defects passed to the next phase or to the field.

This metric is a key concept of the defect removal model for software development.

Maintenance Quality Metrics

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

- Fix backlog and backlog management index
- Fix response time and fix responsiveness

- Percent delinquent fixes
- Fix quality

Fix backlog and backlog management index

Fix backlog is related to the rate of defect arrivals and the rate at which fixes for reported problems become available.

It is a simple count of reported problems that remain at the end of each month or each week.

Using it in the format of a trend chart, this metric can provide meaningful information for managing the maintenance process.

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

$$BMI = \frac{Number\ of\ problems\ closed\ during\ the\ month}{Number\ of\ problems\ arrived\ during\ the\ month} \times 100\%$$

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

Fix response time and fix responsiveness

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

Short fix response time leads to customer satisfaction.

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

Percent delinquent fixes

It is calculated as follows −

Percent Delinquent Fixes =

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

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 records 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 latest 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.

Risk management

Risk Management:

A risk is a probable problem- it might happen or it might not.

There are main two characteristics of risk

Uncertainty: the risk may or may not happen that means there are no 100% risks.

Loss: If the risk occurs in reality, undesirable result or losses will occur.

Risk management is a sequence of steps that help a software team to understand, analyze and manage uncertainty.

Risk management consists of

- Risk Identification
- Risk analysis
- Risk Planning

- Risk Monitoring

A computer code project may be laid low with an outsized sort of risk.

So as to be ready to consistently establish the necessary risks which could have an effect on a computer code project, it's necessary to reason risks into completely different categories.

The project manager will then examine the risks from every category square measure relevant to the project.

There square measure 3 main classes of risks that may have an effect on a computer code project:

- Project Risks
- Technical Risks
- Business Risks

Project Risks:

Project risks concern various sorts of monetary funds, schedules, personnel, resource, and customer-related issues.

A vital project risk is schedule slippage.

Since computer code is intangible, it's terribly tough to observe and manage a computer code project.

It's terribly tough to manage one thing that cannot be seen.

For any producing project, like producing cars, the project manager will see the merchandise taking form.

For example, see that the engine is fitted, at the moment the area of the door unit fitted, the automotive is obtaining painted, etc.

So he will simply assess the progress of the work and manage it.

The physical property of the merchandise being developed is a vital reason why several computer codes come to suffer from the danger of schedule slippage.

Technical Risks:

Technical risks concern potential style, implementation, interfacing, testing, and maintenance issues.

Technical risks conjointly embody ambiguous specifications, incomplete specification, dynamic specification, technical uncertainty, and technical degeneration.

Most technical risks occur thanks to the event team's lean information concerning the project.

Business Risks:

This type of risk embodies the risks of building a superb product that nobody needs, losing monetary funds or personal commitments, etc.

Principle of Risk Management:

- **Global Perspective:** In this, we review the bigger system description, design, and implementation. We look at the chance and the impact the risk is going to have.
- **Take a forward-looking view:** Consider the threat which may appear in the future and create future plans for directing the next events.
- **Open Communication:** This is to allow the free flow of communications between the client and the team members so that they have certainty about the risks.
- **Integrated management:** In this method risk management is made an integral part of project management.
- **Continuous process:** In this phase, the risks are tracked continuously throughout the risk management paradigm.

Reactive Vs Proactive Risk Strategies:

Introduction:

- A "risk" is a situation that could result in a loss or threaten the project's progress but still hasn't happened.
- The process of identifying risks and implementing solutions to limit their impact on the project is known as risk management.
- Risk management's objective is to prevent accidents or substantial losses.
- Reactive risk management aims to minimize the impact of potential dangers and accelerate an Organization's recovery, but it anticipates that threats will occur at some point.
- Proactive risk management seeks to identify risks and mitigate them from arising, to begin with.
- Proactive risk management is a discipline that an organization must practice and embed into its overall business strategy, not a process or a program.
- It can't be defined in a day, and it can't be done alone.
- It is a constant process until it becomes ingrained in the organization's risk culture.

Reactive Risk Management

- **Reactive** risk management tries to reduce the damage of potential threats and speed an organization's recovery from them, but assumes that those threats will happen eventually.

- One fundamental point about reactive risk management is that the disaster or threat must occur before management responds.

  Proactive risk management is all about taking preventative measures before the event to decrease its severity, and that's a good thing to do.

  At the same time, however, organizations should develop reactive risk management plans that can be deployed **after** the event.

  Otherwise management is making decisions about how to respond as the event happens, which can be a costly and stressful ordeal.

  There's an obvious catch-22 with reactive risk management.

  Although this approach gives you time to understand the risk before acting, you're still always one step behind the unfolding threat.

  Other projects will lag as you attend to the problem at hand.

  Helping to Withstand Future Risks

  The reactive approach learns from past or current events and prepares for future events.

  For example, businesses can purchase "cyber Security insurance" to cover the costs of a security disruption.

  This strategy assumes that a breach will happen at some point.

  Once that breach does occur, the business might understand more about how to avoid future breaches, and perhaps could even tailor its insurance policies accordingly.

  Fundamentally, however, the organization reacts **after** the threat has occurred and alters its measures to prevent future potential risks.

  Proactive Risk Management:

  Proactive risk management identifies threats and aims to prevent those events from ever happening in the first place.

  Each strategy has its own activities, metrics, and behaviors that are useful in risk analysis.

  As the name suggests, proactive risk management means that you identify risks before they happen and

figure out ways to avoid or alleviate the risk.

It seeks to reduce the hazard's risk potential or, even better, prevent the threat altogether.

A good example here is vulnerability testing and remediation.

Any organization of appreciable size is likely to have vulnerabilities in its software, which attackers could find an exploit.

So regular testing (or, even better, continuous testing) can help to repair those vulnerabilities and eliminate that particular threat.

Allows for More Control over Risk Management

Proactive management strategy gives you more control over your risk management generally.

You can decide which issues should be top priorities, and what potential damage you're willing to accept.

Proactive management also involves constant monitoring of your systems, risk processes, cyber security, competition, business trends, and so forth.

By understanding the level of risk prior to an event, you can educate and instruct your employees on how to mitigate them.

A truly proactive approach, however, does imply that each risk is constantly monitored.

It also entails regular risk reviews to update the current risk and new risks affecting the company.

This approach drives management to be always aware of the direction of those risks.

Where does predictive risk management fit in all this? As the name suggests, it's all about predicting future risks, outcomes, and threats.

Some predictive components may sound similar to proactive or reactive strategies.

Predictive risk management attempts to:

- Identify probability of risk in a situation, based on one or more variables
- Anticipate potential future risks and their probability
- Anticipate necessary risk controls

What the Prioritization of Risk Management Entails

- Risk identification
- Risk assessment
- Risk treatment (acceptance, avoidance, mitigation, transference)
- Risk monitoring
- Continuous improvement

These are the steps taken with reactive risk management.

This means it's the response to a threat or incident.

It involves:

- Preventing potential risks from becoming incidents
- Mitigating damage from incidents
- Stopping small risks from worsening
- Continuing critical business functions in spite of incidents
- Evaluating each incident to solve its root cause
- Monitoring to ensure that the incident does not recur

Proactive risk management strategies include:

- Identifying existing risks to the enterprise, business unit, or project
- Developing a risk response
- Prioritizing identified risks according to the magnitude of their threat
- Analyzing risks to determine the best treatment for each
- Implementing necessary controls needed to prevent risks from becoming threats or incidents
- Monitoring the threat environment continuously.

Software Risks

- Software risk encompasses the probability of occurrence for uncertain events and their potential for loss within an organization.

- Risk management has become an important component of software development as organizations continue to implement more applications across a multiple technology, multi-tiered environment.
- Typically, software risk is viewed as a combination of robustness, performance efficiency, security and transactional risk propagated throughout the system.

How to Reduce Software Risk:

- Most organizations don't have a process to directly address the software risk that results from active custom software development.
- The traditional approach is to rely on testing – regression tests, system integration tests, performance tests and user integration tests.
- As you can see in the diagram, 30% of defects discovered in QA and live use are structural.
- And it is the structural defects that are the primary software risk exposure in the application lifecycle.
- Based on known software economics, that's 25 defects per function point that directly lead to software risk.
- Adding a structural quality gate to the QA process is imperative in order to measure and prevent software risk in mission critical systems.
- Most structural quality defects are actually **not** related to code quality issues, according to industry sources.
- It's a common misconception that code quality tools might address software risk.
- In reality, structural quality requires system level analysis in order to detect defects that pose software risk.

Why is Software Risk Analysis Imperative?

- Many organizations suffer from failed systems even when a vast amount of time and money are dedicated to functional testing methods.
- The functional approach does identify approximately 90% of the weaknesses that cause system failures; however, it does not account for less apparent issues capable of affecting response times, infrastructure stability, and component functionality issues between application layers.
- Software risk analysis solutions take testing one step further by identifying unknown weaknesses resulting from high severity engineering flaws in multi-tiered systems.
- Analysis solutions designed to locate these issues before execution provide an opportunity to assess potential occurrences and prevent problems before they blatantly become apparent.
- Software risk identification is imperative to business processes in a complex IT environment.
- Proper analysis puts your organization ahead of the curve by allowing for early identification of infrastructure threats and providing the information you need to efficiently manage them.

- Advanced analysis aids in the identification of software risks capable of bringing your entire infrastructure to a screeching halt.
- System wide failures result in lost revenue, customer dissatisfaction, data inconsistencies, and much more.
- Analysis solutions designed to assess business functions as measurable units within an application prevent these types of complications during the development process.
- If your organization is not taking the steps to properly manage these software risk factors in a complex infrastructure, costs or maintenance times resulting from undetected issues could be greatly hindering productivity, performance, or security.

Prevention Is Key!

In a complex technology environment, it is not enough to deal with problems as they become apparent.

Prevention is key to experiencing flawless performance and getting the most out of systems, applications, and your development team.

Exposing the not so obvious weaknesses in an infrastructure by using dependable software risk analysis solutions ensures the proper identification of:

- System Vulnerabilities
- Compliance Issues
- Stability Problems
- Efficiency Weaknesses
- Performance Degradation
- Security Flaws

Are you struggling with pinpointing or managing potential problems in a complex IT environment? Is your organization capable of finding system critical issues prior to executing an application?

CAST offers a dependable solution for early identification and prevention of software risks within a complex, multi-tier environment.

Our Application Intelligence Platform (AIP) can help your organization analyze existing or upcoming deployments to locate and resolve potential issues before they become a bigger problem.

If your organization is seeking a reliable method for assessing risk for complex applications, contact us today to learn more about how software risk analysis can benefit your organization.

Risk Identification:

Methods for Identifying Risks

- Identifying risk is one of most important or essential and initial steps in risk management process.

By chance, if failure occurs in identifying any specific or particular risk, then all other steps that are involved in risk management will not be implemented for that particular risk.

For identifying risk, project team should review scope of program, estimate cost, schedule, technical maturity, parameters of key performance, etc.

To manage risk, project team or organization are needed to know about what risks it faces, and then to evaluate them. Generally, identification of risk is an iterative process. It basically includes generating or creating comprehensive list of threats and opportunities that are based on events that can enhance, prevent, degrade, accelerate, or might delay successful achievement of objectives. In simple words, if you don't find or identify risk, you won't be able to manage it.

The organizer of project needs to expect some of the risk in the project as early as possible so that the performance of risk may be reduced. This could be only possible by making effective risk management planning.

A project may contain large variety of risk. To know the specific amount of risk, there may be chance of affecting a project. So, this is necessary to make categories into different class of risk.

There are many different types of risks which affects the software project:

1. Technology risks

2. Tools risks

3. Estimation risks

4. People risks

5. Requirement risks

6. Organizational risks

Methods for Identifying Risks: Earlier, there were no easy methods available that will surely identify all risks. But nowadays, there are some additional approaches available for identifying risks. Some of approaches for risk identification are given below:

1. Checklist Analysis – Checklist Analysis is type of technique generally used to identify or find risks and manage it. The checklist is basically developed by listing items, steps, or even tasks and is then further analyzed against criteria to just identify and determine if procedure is completed correctly or not. It is list of risk that is just found to occur regularly in development of software project. Below is the list of software development risk by Barry Boehm- modified version.

Risk	Risk Reduction Technique
Personnel Shortfalls	Various techniques include training and career development, job-matching, teambuilding, etc.
Unrealistic time and cost estimates	Various techniques include incremental development, standardization of methods, recording, and analysis of the past project, etc.
Development of wrong software functions	Various techniques include formal specification methods, user surveys, etc.
Development of the wrong user interface	Various techniques include user involvement, prototyping, etc.

2. Brainstorming – This technique provides and gives free and open approach that usually encourages each and everyone on project team to participate. It also results in greater sense of ownership of project risk, and team generally committed to managing risk for given time period of project. It is creative and unique technique to gather risks spontaneously by team members. The team members identify and determine risks in 'no wrong answer' environment. This technique also provides opportunity for team members to always develop on each other's ideas. This technique is also used to determine best possible solution to problems and issue that arises and emerge.

3. Casual Mapping – Causal mapping is method that builds or develops on reflection and review of failure factors in cause and effect of the diagrams. It is very useful for facilitating learning with an organization or system simply as method of project-post evaluation. It is also key tool for risk assessment.

4. SWOT Analysis – Strengths-Weaknesses-Opportunities-Threat (SWOT) is very technique and helpful for identifying risks within greater organization context. It is generally used as planning tool for analyzing business, its resources, and also its environment simply by looking at internal strengths and weaknesses

and opportunities and threats in external environment. It is technique often used in formulation of strategy. The appropriate time and effort should be spent on thinking seriously about weaknesses and threats of organization for SWOT analysis to more effective and successful in risk identification.

5. Flowchart Method – This method allows for dynamic process to be diagrammatically represented in paper. This method is generally used to represent activities of process graphically and sequentially to simply identify the risk.

Risk Projection:

Risk projection, also called risk estimation, attempts to rate each risk in two ways—the likelihood or probability that the risk is real and the consequences of the problems associated with the risk, should it occur.

The project planner, along with other managers and technical staff, performs four risk projection activities:

(1) Establish a scale that reflects the perceived likelihood of a risk.

(2) Delineate the consequences of the risk.

(3) Estimate the impact of the risk on the project and the product.

(4) Note the overall accuracy of the risk projection so that there will be no misunderstandings.

Developing a Risk Table

Risk table provides a project manager with a simple technique for risk projection.

Steps in Setting up Risk Table

(1) Project team begins by listing all risks in the first column of the table. Accomplished with the help of the risk item checklists.

(2) Each risk is categorized in the second column. (e.g. PS implies a project size risk, BU implies a business risk).

(3) The probability of occurrence of each risk is entered in the next column of the table. The probability value for each risk can be estimated by team members individually.

(4) Individual team members are polled in round-robin fashion until their assessment of risk probability begins to converge.

Assessing Risk Impact

Nature of the risk - the problems that are likely if it occurs. e.g. a poorly defined external interface to customer hardware (a technical risk) will preclude early design and testing and will likely lead to system integration problems late in a project.

Scope of a risk - combines the severity with its overall distribution (how much of the project will be affected or how many customers are harmed?).

Timing of a risk - when and how long the impact will be felt.

Overall risk exposure, RE, determined using:

$$\mathbf{RE = P \times C}$$

P is the probability of occurrence for a risk.

C is the cost to the project should the risk occur.

Risk Refinement:

- During early stages of project planning, a risk may be stated quite generally.
- As time passes and more is learned about the project and the risk, it may be possible to refine the risk into a set of more detailed risks, each somewhat easier to mitigate, monitor, and manage.
- One way to do this is to represent the risk in condition-transition-consequence (CTC) format.
- That is, the risk is stated in the following form: Given that <condition> then there is concern that (possibly) <consequence>.
- Using the CTC format for the reuse risk noted in Section 6.4.2, we can write: Given that all reusable software components must conform to specific design standards and that some do not conform, then there is concern that (possibly) only 70 percent of the planned reusable modules may actually be integrated into the as-built system, resulting in the need to custom engineer the remaining 30 percent of components.
- This general condition can be refined in the following manner:

Sub condition 1. Certain reusable components were developed by a third party with no knowledge of internal design standards.

Sub condition 2. The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components.

Sub condition 3. Certain reusable components have been implemented in a language that is not supported on the target environment.

- The consequences associated with these refined sub conditions remains the same (i.e., 30 percent of software components must be customer engineered), but the refinement helps to isolate the underlying risks and might lead to easier analysis and response.

Risk Mitigation, Monitoring, and Management (RMMM):

- A risk management strategy can be defined as a software project plan or the risk management steps.
- It can be organized into a separate Risk Mitigation, Monitoring and Management Plan.
- The RMMM plan documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan.
- Teams do not develop a formal RMMM document. Rather, each risk is documented individually using a risk information sheet.
- In most cases, the RIS is maintained using a database system, so that creation and information entry, priority ordering, searches, and other analysis may be accomplished easily.
- Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence.
- As we have already discussed, risk mitigation is a problem avoidance activity.
- Risk monitoring is a project tracking activity with three primary objectives:

(1) To assess whether predicted risks occur.

(2) To ensure that risk aversion steps defined for the risk are being properly applied; and

(3) To collect information that can be used for future risk analysis.

Effective strategy must consider three issues:

- risk avoidance
- risk monitoring
- Risk management and contingency planning.

Proactive approach to risk – avoidance strategy.

1. Develop risk mitigation plan.

2. Develop a strategy to mitigate this risk for reducing turnover.

3. Meet with current staff to determine causes for turnover.

4. Mitigate those causes that are under our control before the project starts.

- Organize project teams so that information about each development activity is widely dispersed.
- Define documentation standards and establish mechanisms to be sure that documents are developed in a timely manner.
- Project manager monitors for likelihood of risk, Project manager should monitor the effectiveness of risk mitigation steps.
- Risk management and contingency planning assumes that mitigation efforts have failed and that the risk has become a reality.
- RMMM steps incur additional project cost.

Risk Mitigation, Monitoring, and Management (RMMM) plan:

- A risk management technique is usually seen in the software Project plan.
- This can be divided into Risk Mitigation, Monitoring, and Management Plan (RMMM).
- In this plan, all works are done as part of risk analysis.
- As part of the overall project plan project manager generally uses this RMMM plan.
- In some software teams, risk is documented with the help of a Risk Information Sheet (RIS).
- This RIS is controlled by using a database system for easier management of information i.e creation, priority ordering, searching, and other analysis.
- After documentation of RMMM and start of a project, risk mitigation and monitoring steps will start.

Risk Mitigation:

- It is an activity used to avoid problems (Risk Avoidance).
- Steps for mitigating the risks as follows.

1. Finding out the risk.
2. Removing causes that are the reason for risk creation.
3. Controlling the corresponding documents from time to time.
4. Conducting timely reviews to speed up the work.

Risk Monitoring:

- It is an activity used for project tracking.
- It has the following primary objectives as follows.

1. To check if predicted risks occur or not.
2. To ensure proper application of risk aversion steps defined for risk.
3. To collect data for future risk analysis.
4. To allocate what problems are caused by which risks throughout the project.

Risk Management and planning:

- It assumes that the mitigation activity failed and the risk is a reality.
- This task is done by Project manager when risk becomes reality and causes severe problems.
- If the project manager effectively uses project mitigation to remove risks successfully then it is easier to manage the risks.
- This shows that the response that will be taken for each risk by a manager.
- The main objective of the risk management plan is the risk register.
- This risk register describes and focuses on the predicted threats to a software project.

Example:

Let us understand RMMM with the help of an example of high staff turnover.

Risk Mitigation:

To mitigate this risk, project management must develop a strategy for reducing turnover. The possible steps to be taken are:

- Meet the current staff to determine causes for turnover (e.g., poor working conditions, low pay, and competitive job market).
- Mitigate those causes that are under our control before the project starts.
- Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave.
- Organize project teams so that information about each development activity is widely dispersed.

- Define documentation standards and establish mechanisms to ensure that documents are developed in a timely manner.
- Assign a backup staff member for every critical technologist.

Risk Monitoring:

As the project proceeds, risk monitoring activities commence. The project manager monitors factors that may provide an indication of whether the risk is becoming more or less likely. In the case of high staff turnover, the following factors can be monitored:

- General attitude of team members based on project pressures.
- Interpersonal relationships among team members.
- Potential problems with compensation and benefits.
- The availability of jobs within the company and outside it.

Risk Management:

Risk management and contingency planning assumes that mitigation efforts have failed and that the risk has become a reality. Continuing the example, the project is well underway, and a number of people announce that they will be leaving. If the mitigation strategy has been followed, backup is available, information is documented, and knowledge has been dispersed across the team. In addition, the project manager may temporarily refocus resources (and readjust the project schedule) to those functions that are fully staffed, enabling newcomers who must be added to the team to "get up to the speed**".**

Drawbacks of RMMM:

- It incurs additional project costs.
- It takes additional time.
- For larger projects, implementing an RMMM may itself turn out to be another tedious project.
- RMMM does not guarantee a risk-free project, infact, risks may also come up after the project is delivered.

QUALITY MANAGEMENT

What is Software Quality Management?

Software Quality Management ensures that the required level of quality is achieved by submitting improvements to the product development process. SQA aims to develop a culture within the team and it is seen as everyone's responsibility.

Software Quality management should be independent of project management to ensure independence of cost and schedule adherences. It directly affects the process quality and indirectly affects the product quality.

Activities of Software Quality Management:

- **Quality Assurance -** QA aims at developing Organizational procedures and standards for quality at Organizational level.
- **Quality Planning -** Select applicable procedures and standards for a particular project and modify as required to develop a quality plan.
- **Quality Control -** Ensure that best practices and standards are followed by the software development team to produce quality products.

Quality Concepts:

Quality – Quality, as it applies to an object (product, service, process), is defined as the "degree to which a set of inherent characteristics (attributes) of the object satisfies a set of requirements." Therefore, the quality of an object is determined by comparing a predetermined set of characteristics against a set of requirements. If those characteristics conform to the requirements, high quality is achieved, but if those characteristics do not conform, a low or poor level of quality is achieved.

Requirement – A requirement is a need, expectation, or obligation. A specified requirement is one that has been stated, as in the Quality Assurance Project Plan (QAPP) for example. Some of the many types of requirements include those for:

- quality
- quality management
- management

- product
- contracts
- statutes
- regulations

Quality management ensures that an organization, product or service is consistent.

It has four main components:

- Quality planning,
- Quality control
- Quality assurance
- Quality improvement.

Quality management is focused not only on product and service quality, but also the means to achieve it. Quality management therefore uses quality assurance and control of processes as well as products to achieve more consistent quality.

Quality Defined

It is defined as characteristic or attribute of a product or service. Refers to measurable characteristics that we can compare to known standards, in software it involves such measures as cyclomatic complexity, cohesion, coupling, function points, and source lines of code. It includes variation control

- A software development organization should strive to minimize the variation between the predicted and the actual values for cost, schedule, and resources
- They should make sure their testing program covers a known percentage of the software from one release to another
- One goal is to ensure that the variance in the number of bugs is also minimized from one release to another

There are two major types of quality

- Quality of Design: Design quality refers to the level of characteristics that the designers specify for a product.
- Quality of Conformance: The degree to which the design specifications are followed during manufacturing, this focuses on how well the implementation follows the design and how well the resulting system meets its requirements

Software Quality Assurance (SQA):

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 programmer's 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.

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.

Software Review:

Software Review is systematic inspection of software by one or more individuals who work together to find and resolve errors and defects in the software during the early stages of Software Development Life Cycle (SDLC).

Software review is an essential part of Software Development Life Cycle (SDLC) that helps software engineers in validating the quality, functionality and other vital features and components of the software.

It is a whole process that includes testing the software product and it makes sure that it meets the requirements stated by the client.

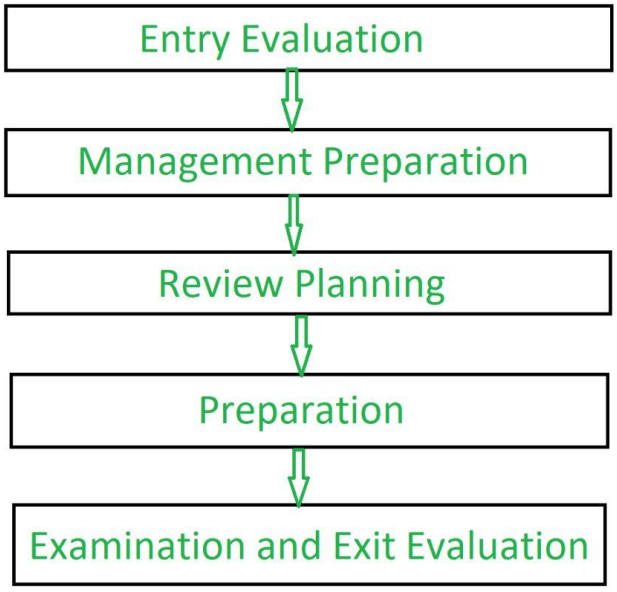
Usually performed manually, software review is used to verify various documents like requirements, system designs, codes, test plans and test cases.

Objectives of Software Review:

The objective of software review is:

1. To improve the productivity of the development team.
2. To make the testing process time and cost effective.
3. To make the final software with fewer defects.
4. To eliminate the inadequacies.

Process of Software Review:



Types of Software Reviews:

There are mainly 3 types of software reviews:

1. **Software Peer Review:** Peer review is the process of assessing the technical content and quality of the product and it is usually conducted by the author of the work product along with some other

developers. Peer review is performed in order to examine or resolve the defects in the software, whose quality is also checked by other members of the team.

Peer Review has following types:

- **(i) Code Review:** Computer source code is examined in a systematic way.
- **(ii) Pair Programming:** It is a code review where two developers develop code together at the same platform.
- **(iii) Walkthrough:** Members of the development team is guided by author and other interested parties and the participants ask questions and make comments about defects.
- **(iv)Technical Review:** A team of highly qualified individuals examines the software product for its client's use and identifies technical defects from specifications and standards.
- **(v) Inspection:** In inspection the reviewers follow a well-defined process to find defects.

2. **Software Management Review:** Software Management Review evaluates the work status. In this section decisions regarding downstream activities are taken.
3. **Software Audit Review:** Software Audit Review is a type of external review in which one or more critics, who are not a part of the development team, organize an independent inspection of the software product and its processes to assess their compliance with stated specifications and standards. This is done by managerial level people.

Advantages of Software Review:

- Defects can be identified earlier stage of development (especially in formal review).
- Earlier inspection also reduces the maintenance cost of software.
- It can be used to train technical authors.
- It can be used to remove process inadequacies that encourage defects.

Formal Technical Review (FTR):

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 observer 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.

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.

Statistical Quality Assurance (SQA)

As brands and retailers experience growing demand for the latest consumer products, the resulting increase in production and batch sizes makes quality control more challenging for companies.

Traditional compliance testing techniques can sometimes provide limited pass/fail information, which results in insufficient measurements on the batch's quality control, identification of the root cause of failure results and overall quality assurance (QA) in the production process.

Intertek combines legal, customer and essential safety requirements to customize a workable QA process, called Statistical Quality Assurance (SQA). SQA is used to identify the potential variations in the manufacturing process and predict potential defects on a parts-per-million (PPM) basis. It provides a statistical description of the final product and addresses quality and safety issues that arise during manufacturing.

SQA consists of three major methodologies:

1. **Force Diagram -** A Force Diagram describes how a product should be tested. Intertek engineers base the creation of Force Diagrams on our knowledge of foreseeable use, critical manufacturing process and critical components that have high potential to fail.

2. **Test-to-Failure (TTF) -** Unlike any legal testing, TTF tells manufacturers on how many defects they are likely to find in every million units of output. This information is incorporated into the process and concludes if a product needs improvement in quality or if it is being over engineered, which will eventually lead to cost savings.

3. **Intervention -** Products are separated into groups according to the total production quantity and production lines. Each group then undergoes an intervention. The end result is measured by Z-value, which is the indicator of quality and consistency of a product to a specification. Intervention allows manufacturers to pinpoint a defect to a specific lot and production line; thus saving time and money in corrective actions.

Quality Assurance:

- Quality Assurance is defined as the auditing and reporting procedures used to provide the stakeholders with data needed to make well-informed decisions.
- It is the Degree to which a system meets specified requirements and customer expectations.
- It is also monitoring the processes and products throughout the SDLC.

Quality Assurance Criteria:

Below are the Quality assurance criteria against which the software would be evaluated against:

- correctness
- efficiency
- flexibility
- integrity

- interoperability
- maintainability
- portability
- reliability
- reusability
- testability
- usability

Software Reliability:

Software Reliability means **Operational reliability**.

It is described as the ability of a system or component to perform its required functions under static conditions for a specific period.

Software reliability is also defined as the probability that a software system fulfills its assigned task in a given environment for a predefined number of input cases, assuming that the hardware and the input are free of error.

Software Reliability is an essential connect of software quality, composed with functionality, usability, performance, serviceability, capability, installability, maintainability, and documentation.

Software Reliability is hard to achieve because the complexity of software turn to be high.

While any system with a high degree of complexity, containing software, will be hard to reach a certain level of reliability, system developers tend to push complexity into the software layer, with the speedy growth of system size and ease of doing so by upgrading the software.

For example, large next-generation aircraft will have over 1 million source lines of software on-board; next-generation air traffic control systems will contain between one and two million lines; the upcoming International Space Station will have over two million lines on-board and over 10 million lines of ground support software; several significant life-critical defense systems will have over 5 million source lines of software.

While the complexity of software is inversely associated with software reliability, it is directly related to other vital factors in software quality, especially functionality, capability, etc.

ISO 9000 Quality Standards

ISO 9000 Certification

ISO (International Standards Organization) is a group or consortium of 63 countries established to plan and fosters standardization. ISO declared its 9000 series of standards in 1987. It serves as a reference for the contract between independent parties. The ISO 9000 standard determines the guidelines for maintaining a quality system. The ISO standard mainly addresses operational methods and organizational methods such as responsibilities, reporting, etc. ISO 9000 defines a set of guidelines for the production process and is not directly concerned about the product itself.

Types of ISO 9000 Quality Standards

The ISO 9000 series of standards is based on the assumption that if a proper stage is followed for production, then good quality products are bound to follow automatically. The types of industries to which the various ISO standards apply are as follows.

1. **ISO 9001:** This standard applies to the organizations engaged in design, development, production, and servicing of goods. This is the standard that applies to most software development organizations.
2. **ISO 9002:** This standard applies to those organizations which do not design products but are only involved in the production. Examples of these category industries contain steel and car manufacturing industries that buy the product and plants designs from external sources and are engaged in only manufacturing those products. Therefore, ISO 9002 does not apply to software development organizations.
3. **ISO 9003:** This standard applies to organizations that are involved only in the installation and testing of the products. For example, Gas companies.

How to get ISO 9000 Certification?

An organization determines to obtain ISO 9000 certification applies to ISO registrar office for registration. The process consists of the following stages:

ISO 9000 Certification

1. **Application:** Once an organization decided to go for ISO certification, it applies to the registrar for registration.
2. **Pre-Assessment:** During this stage, the registrar makes a rough assessment of the organization.
3. **Document review and Adequacy of Audit:** During this stage, the registrar reviews the document submitted by the organization and suggest an improvement.
4. **Compliance Audit:** During this stage, the registrar checks whether the organization has compiled the suggestion made by it during the review or not.
5. **Registration:** The Registrar awards the ISO certification after the successful completion of all the phases.
6. **Continued Inspection:** The registrar continued to monitor the organization time by time.

Advantages of ISO 9000 Certification:

Some of the advantages of the ISO 9000 certification process are following :

- Business ISO-9000 certification forces a corporation to specialize in “how they are doing business”. Each procedure and work instruction must be documented and thus becomes a springboard for continuous improvement.
- Employees morale is increased as they’re asked to require control of their processes and document their work processes
- Better products and services result from continuous improvement process.
- Increased employee participation, involvement, awareness and systematic employee training are reduced problems.

Shortcomings of ISO 9000 Certification:

Some of the shortcoming of the ISO 9000 certification process are following :

- ISO 9000 does not give any guideline for defining an appropriate process and does not give guarantee for high quality process.
- ISO 9000 certification process have no international accreditation agency exists.

Overview:

The mission of the International Standard Organization is to market the development of standardization and its related activities to facilitate the international exchange of products and services. It also helps to develop cooperation within the different activities like spheres of intellectual, scientific, technological, and economic activity.

ISO 9000 Quality Standards:

It is defined as the quality assurance system in which quality components can be organizational structure, responsibilities, procedures, processes, and resources for implementing quality management. Quality assurance systems are created to help organizations ensure their products and services satisfy customer expectations by meeting their specifications.

Different types of ISO standards:

Here, you will see different types of ISO standards as follows.

- **ISO 9000: 2000 –** ISO 9000: 2000: contains Quality management systems, fundamentals, and vocabulary.
- **ISO 9000-1: 1994 –**This series of standards includes Quality management systems and Quality assurance standards. It also includes some guidelines for selection and use.
- **ISO 9000-2: 1997 –**This series of standards also includes Quality management systems and Quality assurance standards. It also includes some guidelines for the application of ISO 9001, ISO 9002, and ISO 9003.

- **ISO 9000-3: 1997 –**This series contains Quality management systems, Quality assurance standards and also includes guidelines for the application of ISO 9001 to 1994 to the development, supply, installation, and maintenance of computer installation.

- **ISO 9001**: **1994** –This series of standards has Quality systems and a Quality assurance model. This model helps in design, development, production, installation, and service.

- **ISO 9001**: **2000 –**This series of standards also includes Quality management systems.

- **ISO 9002**: **1994 –**This series of standards also includes some Quality systems. This Quality assurance model used in production, installation, and servicing.

- **ISO 9003**: **1994 –**This series of standards also includes some Quality systems. This Quality assurance model used in the final inspection and test.

- **ISO 9004**: **2000 –**This series of standards include some Quality management systems. It also includes some guidelines for performance improvements.

- **ISO 9039: 1994 –**This series of standards include some Optics and Optical Instruments. It includes quality evaluation of optical systems and determination of distortion.

- **ISO/IEC 9126-1: 2001 –**This series of standards has information technology. It also includes some software products, quality models.

- **ISO/IEC 9040: 1997 –**This series of standards has information technology. It also includes open system interconnection and Virtual terminal basic class service.

- **ISO/IEC 9041-1: 1997 –**This series of standards has information technology. It also includes open system interconnection, Virtual terminal basic class service protocol, and specification.

- **ISO/IEC 9041-2: 1997 –**This series of standards include information technology, open system interconnection, Virtual terminal basic class protocol, and Protocol implementation conformance statement (PICS) Performa.

- **ISO/IEC 9075-9: 2001 –**This series of standards has information technology, Database languages, and SQL/MED(Management of External Data).

- **ISO/IEC 9075-10: 2000 -**This series of standards has information technology, Database languages, and SQL/OLB (Object Language Bindings).

- **ISO/IEC 9075-13: 2002 –**This series of standards has information technology, Database languages, SQL routines, and Java Programming language. (SQL/JRT).