Module 5

Risk Management

- Risk analysis and management are a series of steps that help a software team to understand and manage uncertainty.
- 1.Risk identification
- 2. Risk Projection
- 3. Risk assessment
- 4. Risk management

- Risk always involves two characteristics a set of risk information sheets is produced.
- ***Uncertainty***—the risk may or may not happen; that is, there are no 100% probable risks.
- ***Loss***—if the risk becomes a reality, unwanted consequences or losses will occur.

- ***Project risks*** threaten the project plan. That is, if project risks become real, it is likely that project schedule will slip and that costs will increase. Project risks identify potential budgetary, schedule, personnel (staffing and organization), resource, customer, and
- requirements problems and their impact on a software project.
- ***Technical risks*** threaten the quality and timeliness of the software to be produced. If a technical risk becomes a reality, implementation may become difficult or impossible.
- ***Business risks*** threaten the viability of the software to be built. Business risks often jeopardize the project or the product. Candidates for the top five business risks are
- Market risk,
- Strategic risk,
- Management risk, and
- Budget risks.

- ***Known risks*** are those that can be uncovered after careful evaluation of the project plan.
- ***Predictable risks*** are extrapolated from past project experience (e.g., staff turnover, poor
- communication with the customer, dilution of staff effort as ongoing maintenance
- requests are serviced).
- ***Unpredictable risks*** are the joker in the deck. They can and do occur, but they are
- extremely difficult to identify in advance.

- **Reactive risk strategies:-**
- 1. Reactive risk strategies follows that the risks have to be tackled at the time of their occurrence.
- 2. No precautions are to be taken as per this strategy.
- 3. They are meant for risks with relatively smaller impact.
- **Proactive risk strategies:-**
- 1. Proactive risk strategies follows that the risks have to be identified before start of the project.
- 2. They have to be analyzed by assessing their probability of occurrence, their impact after occurrence, and steps to be followed for its precaution.
- 3. They are meant for risks with relatively higher impact.

Risk identification

- ***Risk identification*** is a systematic attempt to specify threats to the project plan
- One method for identifying risks is to create a **risk item checklist.**

- ***Product size***—risks associated with the overall size of the software to be built or modified.
- ***Business impact***—risks associated with constraints imposed by management or the marketplace.
- ***Customer characteristics***—risks associated with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.
- ***Process definition***—risks associated with the degree to which the software process has been defined and is followed by the development organization.
- ***Development environment***—risks associated with the availability and quality of the tools to be used to build the product.
- ***Technology to be built***—risks associated with the complexity of the system to be built and the "newness" of the technology that is packaged by the system.
- ***Staff size and experience***—risks associated with the overall technical and project experience of the software engineers who will do the work.

I. Assessing Overall Project Risk

- Is the software project we're working on at serious risk?
- The questions are ordered by their relative importance to the success of a project.
- 1. Have top software and customer managers formally committed to support the project?
- 2. Are end-users enthusiastically committed to the project and the system/product to be built?
- 3. Are requirements fully understood by the software engineering team and their customers?
- 4. Have customers been involved fully in the definition of requirements?

- 5. Do end-users have realistic expectations?
- 6. Is project scope stable?
- 7. Does the software engineering team have the right mix of skills?
- 8. Are project requirements stable?
- 9. Does the project team have experience with the technology to be implemented?
- 10. Is the number of people on the project team adequate to do the job?
- 11. Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?
- If any one of these questions is answered negatively, mitigation, monitoring, and management steps should be instituted without fail.

II. Risk Components and Drivers

- The risk components are defined in the following manner:
- **Performance** risk—the degree of uncertainty that the product will meet its requirements and be fit for its intended use.
- **Cost** risk—the degree of uncertainty that the project budget will be maintained.
- **Support** risk—the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.
- **Schedule** risk—the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.
- The impact of each risk driver on the risk component is divided into one of four impact categories—**negligible, marginal, critical, or catastrophic.**

Impact assessment

Components / Category		Performance	Support	Cost	Schedule
Catastrophic	1	Failure to meet the requirement would result in mission failure		Failure results in increased costs and schedule delays with expected values in excess of $500K	
	2	Significant degradation to nonachievement of technical performance	Nonresponsive or unsupportable software	Significant financial shortages, budget overrun likely	Unachievable IOC
Critical	1	Failure to meet the requirement would degrade system performance to a point where mission success is questionable		Failure results in operational delays and/or increased costs with expected value of $100K to $500K	
	2	Some reduction in technical performance	Minor delays in software modifications	Some shortage of financial resources, possible overruns	Possible slippage in IOC
Marginal	1	Failure to meet the requirement would result in degradation of secondary mission		Costs, impacts, and/or recoverable schedule slips with expected value of $1K to $100K	
	2	Minimal to small reduction in technical performance	Responsive software support	Sufficient financial resources	Realistic, achievable schedule
Negligible	1	Failure to meet the requirement would create inconvenience or nonoperational impact		Error results in minor cost and/or schedule impact with expected value of less than $1K	
		No reduction in	Easily supportable	Possible budget	Early

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. Establishing a scale that reflects the perceived likelihood of a risk.
- 2. Delineating the consequences of the risk.
- 3. Estimating the impact of the risk of the project and the product.
- 4. Noting the overall accuracy of the risk projection so that there will be no misunderstandings.

I. Developing a Risk Table

Risks	Category	Probability	Impact	RMMM
Size estimate may be significantly low	PS	60%	2	
Larger number of users than planned	PS	30%	3	
Less reuse than planned	PS	70%	2	
End-users resist system	BU	40%	3	
Delivery deadline will be tightened	BU	50%	2	
Funding will be lost	CU	40%	1	
Customer will change requirements	PS	80%	2	
Technology will not meet expectations	TE	30%	1	
Lack of training on tools	DE	80%	3	
Staff inexperienced	ST	30%	2	
Staff turnover wil be high	ST	60%	2	
•				
•				
•				

Impact values:

For example, Assume that the software team defines a project risk in the following manner-

- **Risk identification. Only 70 percent of the software components scheduled for reuse** will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.
- **Risk probability. 80% (likely).**
- **Risk impact. 60 reusable software components were planned. If only 70 percent can be** used, 18 components would have to be developed from scratch (in addition to other custom software that has been scheduled for development). Since the average component is 100 LOC and local data indicate that the software engineering cost for each LOC is $14.00,
- the overall cost (impact) to develop the components would be 18 x 100 x 14 = $25,200.
- **Risk exposure. *RE* = *0.80 x 25,200 ~ $20,200.***

RISK MITIGATION, MONITORING, AND MANAGEMENT

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

- **risk mitigation** is a problem avoidance activity.
- **Risk monitoring** is a project tracking activity with three primary objectives:
- (1) to assess whether predicted risks do, in fact, 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.

Risk information sheet

Risk ID: P02-4-32	Date: 5/9/02	Prob: 80%	Impact: high

Description:
Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.

Refinement/context:
Subcondition 1: Certain reusable components were developed by a third party with no knowledge of internal design standards.
Subcondition 2: The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components.
Subcondition 3: Certain reusable components have been implemented in a language that is not supported on the target environment.

Mitigation/monitoring:
1. Contact third party to determine conformance with design standards.
2. Press for interface standards completion; consider component structure when deciding on interface protocol.
3. Check to determine number of components in subcondition 3 category; check to determine if language support can be acquired.

Management/contingency plan/trigger:
RE computed to be $20,200. Allocate this amount within project contingency cost. Develop revised schedule assuming that 18 additional components will have to be custom built; allocate staff accordingly.
Trigger: Mitigation steps unproductive as of 7/1/02

Current status:
5/12/02: Mitigation steps initiated.

Originator: D. Gagne	Assigned: B. Laster

SOFTWARE CONFIGURATION MANAGEMENT (SCM)

SOFTWARE CONFIGURATION MANAGEMENT

- The output of the software process is information that may be divided into three broad categories:

(1) computer programs (both source level and executable forms);

(2) documents that describe the computer programs (targeted at both technical practitioners and users), and

(3) data.

There are four fundamental sources of change:

- New business or market conditions dictate changes in product requirements or business rules.
- New customer needs demand modification of data produced by information systems,
- Reorganization or business growth/downsizing causes changes in project priorities or software engineering team structure.
- Budgetary or scheduling constraints.

- Free software tools that help in SCM are:
  - Concurrent Versions System (CVS)
  - Revision Control System (RCS)
  - Source Code Control System (SCCS)
- Commercial Tools:
  - Rational Clear Case
  - PVCS
  - Microsoft Visual SourceSafe

Baselines

- A *baseline is a software configuration management concept that helps us to control.*
- Change without seriously impeding justifiable change.

Five SCM tasks

- **Identification** (tracking multiple versions to enable efficient changes)
- **Version control** (control changes before and after release to customer)
- **Change control** (authority to approve and prioritize changes)
- **Configuration auditing** (ensure changes made properly)
- **Reporting** (tell others about changes made)

Difference between SCM and Software Support (Maintenance)

- Support is a set of software engineering activities that occur after software has been delivered to the customer and put into operation.
- Software configuration management is a set of tracking and control activities that begin when a software engineering project begins and terminate only when the software is taken out of operation.

- A configuration object has a name, attributes, and is "connected" to

FIGURE
Configuration objects

Version Control

- Configuration management allows a user to specify alternative configurations of the software system through the selection of appropriate versions
- This is supported by associating attributes with each software version, and then allowing a configuration to be specified [and constructed] by describing the set of desired attributes.
- These "attributes" mentioned can be as simple as a specific version number that is attached to each object or as complex as a string of Boolean variables

FIGURE

Evolution graph

Change Control

- Change request is submitted and evaluated to assess technical merit and impact on the other configuration objects and budget.
- Change report contains the results of the evaluation.
- Change control authority (CCA) makes the final decision on the status and priority of the change based on the change report.
- Engineering change order (ECO) is generated for each change approved .
- Object to be changed is checked-out of the project database subject to access control parameters for the object.
- Modified object is subjected to appropriate SQA and testing procedures.
- Modified object is checked-in to the project database and version control mechanisms are used to create the next version of the software.
- Synchronization control is used to ensure that parallel changes made by different people don't overwrite one another.

FIGURE
The change control process

Configuration Audit

- To ensure that the change has been properly implemented we conduct :
  - formal technical reviews and
  - the software configuration audit.
- The **formal technical review** focuses on the technical correctness of the configuration object that has been modified.
- A **software configuration audit** complements the formal technical review by assessing a configuration object for characteristics that are generally not considered during review.

- The audit asks and answers the following questions:
  - Has the change specified in the ECO (Engineering change order) been made? Have any additional modifications been incorporated?
  - Has a formal technical review been conducted to assess technical correctness?
  - Has the software process been followed and have software engineering standards been properly applied?
  - Has the change been "highlighted" in the SCI? Have the change date and change author been specified? Do the attributes of the configuration object reflect the change?
  - Have SCM procedures for noting the change, recording it, and reporting it been followed?
  - Have all related SCIs been properly updated?

Status Reporting

- Configuration status reporting (sometimes called status accounting) is an SCM task that answers the following questions:
  - What happened?
  - Who did it?
  - When did it happen?
  - What else will be affected?
    - Each time an SCI is assigned new or updated identification, a CSR entry is made.
    - Each time a configuration audit is conducted, the results are reported as part of the CSR task.
    - Output from CSR may be placed in an on-line database.

Software Quality Assurance

What is Quality?

- Quality defines to any measurable characteristics such as correctness, maintainability, portability, testability, usability, reliability, efficiency, integrity, reusability, and interoperability

There are two kinds of Quality:

- **Quality of Design:** Quality of Design refers to the characteristics that designers specify for an item. The grade of materials, tolerances, and performance specifications that all contribute to the quality of design.
- **Quality of conformance:** Quality of conformance is the degree to which the design specifications are followed during manufacturing. Greater the degree of conformance, the higher is the level of quality of conformance.
- **Software Quality:** Software Quality is defined as the conformance to explicitly state functional and performance requirements, explicitly documented development standards, and inherent characteristics that are expected of all professionally developed software.

- **Quality Control:** Quality Control involves a series of inspections, reviews, and tests used throughout the software process to ensure each work product meets the requirements place upon it. Quality control includes a feedback loop to the process that created the work product.
- **Quality Assurance:** Quality Assurance is the preventive set of activities that provide greater confidence that the project will be completed successfully.
- **Quality Assurance** focuses on how the engineering and management activity will be done?
- As anyone is interested in the quality of the final product, it should be assured that we are building the right product.
- It can be assured only when we do inspection & review of intermediate products, if there are any bugs, then it is debugged. This quality can be enhanced

Importance of Quality

- We would expect the quality to be a concern of all producers of goods and services. However, the distinctive characteristics of software and in particular its intangibility and complexity, make special demands.
- **Increasing criticality of software:** The final customer or user is naturally concerned about the general quality of software, especially its reliability. This is increasing in the case as organizations become more dependent on their computer systems and software is used more and more in safety-critical areas. For example, to control aircraft.

- **The intangibility of software:** This makes it challenging to know that a particular task in a project has been completed satisfactorily. The results of these tasks can be made tangible by demanding that the developers produce 'deliverables' that can be examined for quality.
- **Accumulating errors during software development:** As computer system development is made up of several steps where the output from one level is input to the next, the errors in the earlier ?deliverables? will be added to those in the later stages leading to accumulated determinable effects. In general the later in a project that an error is found, the more expensive it will be to fix. In addition, because the number of errors in the system is unknown, the debugging phases of a project are particularly challenging to contro

Quality Management

Quality Concepts

- Variation control is the heart of quality control
- Form one project to another, we want to minimize the difference between the predicted resources needed to complete a project and the actual resources used, including staff, equipment, and calendar time
- Quality of design

  Refers to characteristics that designers specify for the end product

Quality Management

- Quality of conformance

  Degree to which design specifications are followed in manufacturing the product
- Quality control

  Series of inspections, reviews, and tests used to ensure conformance of a work product to its specifications
- Quality assurance

  Consists of a set of auditing and reporting functions that assess the effectiveness and completeness of quality control activities

Quality Management

- **Cost of Quality**
- Prevention costs

  Quality planning, formal technical reviews, test equipment, training
- Appraisal costs

  In-process and inter-process inspection, equipment calibration and maintenance, testing
- Failure costs

  rework, repair, failure mode analysis
- External failure costs

  Complaint resolution, product return and replacement, help line support, warranty work

Software Quality Assurance

- Software quality assurance (SQA) is the concern of
- every software engineer to reduce cost and improve
- product time-to-market.
- A Software Quality Assurance Plan is not merely
- another name for a test plan, though test plans are
- included in an SQA plan.
- SQA activities are performed on every software
- project.
- Use of metrics is an important part of developing a
- strategy to improve the quality of both software
- processes and work products.

Software Quality Assurance

- Software quality assurance is a planned and systematic plan of all actions necessary to provide adequate confidence that an item or product conforms to establish technical requirements.
- A set of activities designed to calculate the process by which the products are developed or manufactured.
- SQA Encompasses
  - A quality management approach
  - Effective Software engineering technology (methods and tools)
  - Formal technical reviews that are tested throughout the software process
  - A multitier testing strategy
  - Control of software documentation and the changes made to it.
  - A procedure to ensure compliances with software development standards
  - Measuring and reporting mechanisms.

SQA Activities

- Software quality assurance is composed of a variety of functions associated with two different constituencies ? the software engineers who do technical work and an SQA group that has responsibility for quality assurance planning, record keeping, analysis, and reporting.

- **Following activities are performed by an independent SQA group:**

1. **Prepares an SQA plan for a project:** The program is developed during project planning and is reviewed by all stakeholders. The plan governs quality assurance activities performed by the software engineering team and the SQA group. The plan identifies calculation to be performed, audits and reviews to be performed, standards that apply to the project, techniques for error reporting and tracking, documents to be produced by the SQA team, and amount of feedback provided to the software project team.
2. **Participates in the development of the project's software process description:** The software team selects a process for the work to be performed. The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards (e.g. ISO-9001), and other parts of the software project plan.
3. **Reviews software engineering activities to verify compliance with the defined software process:** The SQA group identifies, reports, and tracks deviations from the process and verifies that corrections have been made.
4. **Audits designated software work products to verify compliance with those defined as a part of the software process:** The SQA group reviews selected work products, identifies, documents and tracks deviations, verify that corrections have been made, and periodically reports the results of its work to the project manager.
5. **Ensures that deviations in software work and work products are documented and handled according to a documented procedure:** Deviations may be encountered in the project method, process description, applicable standards, or technical work products.
6. **Records any noncompliance and reports to senior management:** Non- compliance items are tracked until they are resolved.

Quality Assurance v/s Quality control

Quality Assurance	Quality Control
Quality Assurance (QA) is the set of actions including facilitation, training, measurement, and analysis needed to provide adequate confidence that processes are established and continuously improved to produce products or services that conform to specifications and are fit for use.	**Quality Control (QC)** is described as the processes and methods used to compare product quality to requirements and applicable standards, and the actions are taken when a nonconformance is detected.
QA is an activity that establishes and calculates the processes that produce the product. If there is no process, there is no role for QA.	**QC** is an activity that demonstrates whether or not the product produced met standards.
QA helps establish process	**QC** relates to a particular product or service
QA sets up a measurement program to evaluate processes	**QC** verified whether particular attributes exist, or do not exist, in a explicit product or service.
QA identifies weakness in processes and improves them	**QC** identifies defects for the primary goals of correcting errors.
Quality Assurance is a managerial tool.	Quality Control is a corrective tool.
Verification is an example of QA.	Validation is an example of QC.

Software Reviews

- Purpose is to find errors before they are
- passed on to another software engineering
- activity or released to the customer.
- Software engineers (and others) conduct
- formal technical reviews (FTRs) for software
- quality assurance.
- Using formal technical reviews (walkthroughs
- or inspections) is an effective means for
- improving software quality.

Formal Technical Review

- A FTR is a software quality control activity performed by software engineers and others.The objectives are:

1. To uncover errors in function, logic or implementation for any representation of the software.
2. To verify that the software under review meets its requirements.
3. To ensure that the software has been represented according to predefined standards.
4. To achieve software that is developed in a uniform manner and
5. To make projects more manageable.

Formal Technical Review

Review meeting in FTR

- The Review meeting in a FTR should abide to the following constraints

1. Review meeting members should be between three and five.
2. Every person should prepare for the meeting and should not require more than two hours of work for each person.
3. The duration of the review meeting should be less than two hours.

Formal Technical Review

- The focus of FTR is on a work product that is requirement specification, a detailed component design, a source code listing for a component.
- The individual who has developed the work product i.e, the producer informs the project leader that the work product is complete and that a review is required.
- The project leader contacts a review leader, who evaluates the product for readiness, generates copy of product material and distributes them to two or three review members for advance preparation .
- Each reviewer is expected to spend between one and two hours reviewing the product, making notes

Formal Technical Review

- The review leader also reviews the product and establish an agenda for the review meeting
- The review meeting is attended by review leader, all reviewers and the producer.
- One of the reviewer act as a recorder,who notes down all important points discussed in the meeting.
- The meeting(FTR) is started by introducing the agenda of meeting and then the producer introduces his product. Then the producer “walkthrough” the product, the reviewers raise issues which they have prepared in advance.
- If errors are found the recorder notes down

Review reporting and Record keeping

- During the FTR, a reviewer(recorder) records all issues that have been raised
- A review summary report answers three questions
  1. What was reviewed?
  2. Who reviewed it?
  3. What were the findings and conclusions?
- Review summary report is a single page form with possible attachments
- The review issues list serves two purposes
  1. To identify problem areas in the product
  2. To serve as an action item checklist that guides the producer as corrections are made

Review Guidelines

- Review the product, not the producer
- Set an agenda and maintain it
- Limit debate and rebuttal
- Enunciate problem areas, but don’t attempt to solve every problem noted
- Take return notes
- Limit the number of participants and insist upon advance preparation.
- Develop a checklist for each product i.e likely to be reviewed
- Allocate resources and schedule time for FTRS
- Conduct meaningful training for all reviewer • Review your early reviews

Software Defects

- Industry studies suggest that design activities
- introduce 50-65% of all defects or errors
- during the software process
- Review techniques have been shown to be up
- to 75% effective in uncovering design flaws
- which ultimately reduces the cost of • subsequent activities in the software process

Software Quality Metrices

- In Software Engineering, Software Measurement is done based on some Software Metrics where these software metrics are referred to as the measure of various characteristics of a Software.
- In Software engineering Software Quality Assurance (SAQ) assures the quality of the software. A set of activities in SAQ is continuously applied throughout the software process. **Software Quality is measured based on some software quality metrics.**
- **There is a number of metrics available based on which software quality is measured**. But among them, there are a few most useful metrics which are essential in software quality measurement. They are –
  - ✓ Code Quality
  - ✓ Reliability
  - ✓ Performance
  - ✓ Usability
  - ✓ Correctness
  - ✓ Maintainability
  - ✓ Integrity
  - ✓ Security

Software Quality Metrices

1. Code Quality – Code quality metrics measure the quality of code used for software project development. Maintaining the software code quality by writing Bug-free and semantically correct code is very important for good software project development. In code quality, both Quantitative metrics like the number of lines, complexity, functions, rate of bugs generation, etc, and Qualitative metrics like readability, code clarity, efficiency, and maintainability, etc are measured.

2. Reliability – Reliability metrics express the reliability of software in different conditions. The software is able to provide exact service at the right time or not checked. Reliability can be checked using Mean Time Between Failure (MTBF) and Mean Time To Repair (MTTR).

3. Performance – Performance metrics are used to measure the performance of the software. Each software has been developed for some specific purposes. Performance metrics measure the performance of the software by determining whether the software is fulfilling the user requirements or not, by analyzing how much time and resource it is utilizing for providing the service.

Software Quality Metrices

4. Usability – Usability metrics check whether the program is user-friendly or not. Each software is used by the end-user. So it is important to measure that the end-user is happy or not by using this software.

5. Correctness – Correctness is one of the important software quality metrics as this checks whether the system or software is working correctly without any error by satisfying the user. Correctness gives the degree of service each function provides as per developed.

6. Maintainability – Each software product requires maintenance and up-gradation. Maintenance is an expensive and time-consuming process. So if the software product provides easy maintainability then we can say software quality is up to mark. Maintainability metrics include the time required to adapt to new features/functionality, Mean Time to Change (MTTC), performance in changing environments, etc.

Software Quality Metrices

7. Integrity – Software integrity is important in terms of how much it is easy to integrate with other required software which increases software functionality and what is the control on integration from unauthorized software's which increases the chances of cyberattacks.

8. Security – Security metrics measure how secure the software is. In the age of cyber terrorism, security is the most essential part of every software. Security assures that there are no unauthorized changes, no fear of cyber attacks, etc when the software product is in use by the end-user.

McCall's Software Quality Model

- **McCall's Software Quality Model** was introduced in 1977. This model is incorporated with many attributes, termed as software factors, which influence software. The model distinguishes between two levels of quality attributes:
  - Quality Factors
  - Quality Criteria
- **Quality Factors:** The higher-level quality attributes which can be accessed directly are called quality factors. These attributes are external attributes. The attributes at this level are given more importance by the users and managers.
- **Quality Criteria:** The lower or second-level quality attributes that can be accessed either subjectively or objectively are called Quality Criteria. These attributes are internal attributes. Each quality factor has many second-level of quality attributes or quality criteria.

- The usability quality factor is divided into training, communicativeness, input/output volume, input/output rate. This model classifies all software requirements into 11 software quality factors. The factors are organized into three product quality factors: **Product Operation**, **Product Revision**, and **Product Transition.**

Factors of Product Quality

- **Product Operation**
- It includes five software quality factors, which are related to the requirements that directly affect the operation of the software such as operational performance, convenience, ease of usage, and correctness. These factors help in providing a better user experience.
- **Correctness:** The extent to which software meets its requirements specification.
- **Efficiency:** The number of hardware resources and code the software, needs to perform a function.
- **Integrity:** The extent to which the software can control an unauthorized person from accessing the data or software.
- **Reliability:** The extent to which software performs its intended functions without failure.
- **Usability:** The extent of effort required to learn, operate and understand the functions of the software.

Factors of Product Quality

- **Product Revision**
- It includes three software quality factors, which are required for testing and maintenance of the software. They provide ease of maintenance, flexibility, and testing effort to support the software to be functional according to the needs and requirements of the user in the future.
- **Maintainability:** The effort required to detect and correct an error during the maintenance phase.
- **Flexibility:** The effort needed to improve an operational software program.
- **Testability:** The effort required to verify software to ensure that it meets the specified requirements.

Factors of Product Quality

- **Product Transition**
- It includes three software quality factors, that allow the software to adapt to the change of environments in the new platform or technology from the previous.
- **Portability:** The effort required to transfer a program from one platform to another.
- **Re-usability:** The extent to which the program's code can be reused in other applications.
- **Interoperability:** The effort required to integrate two systems with one another.

Software Reliability

- Defined as the probability of failure free operation of a
- computer program in a specified environment for a specified
- time period
- Can be measured directly and estimated using historical and
- developmental data
- Software reliability problems can usually be traced back to
- errors in design or implementation.
- Measures of Reliability
- Mean time between failure (MTBF) = MTTF + MTTR
- MTTF = mean time to failure
- MTTR = mean time to repair
- Availability = [MTTF / (MTTF + MTTR)] x 100%

ISO 9000 Quality Standards

- Quality assurance systems are defined as the
- organizational structure, responsibilities, procedures,
- processes, and resources for implementing quality
- management.
- ISO 9000 describes the quality elements that must
- be present for a quality assurance system to be
- compliant with the standard, but it does not describe
- how an organization should implement these
- elements.
- ISO 9001:2000 is the quality standard that contains
- 20 requirements that must be present in an effective
- software quality assurance system.

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:

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.