UNIT-V

SHORT Q&A:

1.Define reactive and proactive risk strategies

Reactive: "A response based risk management approach, which is dependent on accident E2vaIuation and audit based findings."

Proactive: "Adaptive, closed loop feedback control **strategy** based on measurement, observation of the present safety level and planned explicit target safety level with a creative intellectuality.

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

3.Define the formulae for measures of reliability and availability.

Ans: Software reliability is defined in statistical terms as "the probability of failure-free operation of a computer program in a specified environment for a specified time"

If we consider a computer-based system, a simple measure of reliability is meantime-between-failure (MTBF), ***where MTBF = MTTF + MTTR***

The acronyms MTTF and MTTR are mean-time-to-failure and mean-time-to-repair, respectively.

Software availability is the probability that a program is operating according to requirements at a given point in time and is defined as

$$Availability = [MTTF/(MTTF + MTTR)] * 100\%$$

4.What are the types of software risks?

Ans: 1. Project risk
2. Technical Risk
3. Business Risk

5.What is the importance of software reviews?

Important reasons for Software Review are: It improves the productivity of the development team. Makes the process of testing time & cost effective, as more time is spent

on testing the software during the initial development of the product.

6.Define software reliability?

Software Reliability is defined as: the probability of failure-free software operation for a specified period of time in a specified environment.

7.Define SQA plan?

Ans: • Software Quality can be defined as Conformance to explicitly state functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.

- The plan identifies

•Evaluations to be performed.

•Audits and reviews to be performed.

•Standards that is applicable to the project.

•Procedures for error reporting and tracking.

•Documents to be produced by the SQA group.

•Amount of feedback provided to the software project team.

8. What does RMMM stands in RMMM plan.

Ans. • The RMMM (RISK MITIGATION, MONITORING, AND MANAGEMENT)plan documents all work performed as part of risk analysis and are used by the project manager as part of the overall project plan.

•Some software teams do not develop a formal RMMM document. Rather, each risk is documented individually using a Risk Information Sheet (RIS).

•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.

•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.

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

9. Give a short notes on formal technical reviews.

- To uncover errors in function, logic, or implementation for any representation of the software;
- To verify that the software under **review** meets its requirements;
- To ensure that the software has been represented according to predefined standards.
- To achieve software that is developed in a uniform manner;

10.Describe six sigma for software.

Six Sigma is the process of improving the quality of the output by identifying and eliminating the cause of defects and reduce variability in manufacturing and business processes. The maturity of a manufacturing process can be defined by a **sigma** rating indicating its percentage of defect-free products it creates.

LONG Q&A:

1.Explain about Quality concepts? Describe metrics for source code and for testing

Software quality product is defined in term of its fitness of purpose. That is, a quality product does precisely what the users want it to do. For software products, the fitness of use is generally explained in terms of satisfaction of the requirements laid down in the SRS document. Although "fitness of purpose" is a satisfactory interpretation of quality for many devices such as a car, a table fan, a grinding machine, etc.for software products, "fitness of purpose" is not a wholly satisfactory definition ofquality.

Example: Consider a functionally correct software product. That is, it performs all tasks as specified in the SRS document. But, has an almost unusable user interface. Even though it may be functionally right, we cannot consider it to be a quality product.

The modern view of a quality associated with a software product **several** quality **methods such as the following:**

Portability: A software device is said to be portable, if it can be freely made to work in various operating system environments, in multiple machines, with other software products, etc.

Usability: A software product has better usability if various categories of users can easily invoke the functions of the product.

Reusability: A software product has excellent reusability if different modules of the product can quickly be reused to develop new products.

Correctness: A software product is correct if various requirements as specified in the SRS document have been correctly implemented.

Maintainability: A software product is maintainable if bugs can be easily corrected as and when they show up, new tasks can be easily added to the product, and the functionalities of the product can be easily modified, etc.

metrics for source code

Halstead's theory of "software science" [Hal77] proposed the first analytical "laws" for computer software.14 Halstead assigned quantitative laws to the development of computer software, using a set of primitive measures that may be derived after code is generated or estimated once design is complete. The measures are: n1 number of distinct operators that appear in a program n2 number of distinct operands that appear

in a program N1 total number of operator occurrences N2 total number of operand occurrences Halstead uses these primitive measures to develop expressions for the overall program length, potential minimum volume for an algorithm, the actual volume (number of bits required to specify a program), the program level (a measure of software complexity), the language level (a constant for a given language), and other features such as development effort, development time, and even the projected number of faults in the software.

METRICS FOR TESTING

Although much has been written on software metrics for testing (e.g., [Het93]), the majority of metrics proposed focus on the process of testing, not the technical characteristics of the tests themselves. In general, testers must rely on analysis, design, and code metrics to guide them in the design and execution of test cases. Architectural design metrics provide information on the ease or difficulty associated with integration testing (Section 23.3) and the need for specialized testing software (e.g., stubs and drivers). Cyclomatic complexity (a component-level design metric) lies at the core of basis path testing, a test-case design method presented in Chapter 18. In addition, cyclomatic complexity can be used to target modules as candidates for extensive unit testing. Modules with high cyclomatic complexity are more likely to be error prone than modules whose cyclomatic complexity is lower. For this reason, you should extend above average effort to uncover errors in such modules before they are integrated in a system.

2.Explain about the importance of test strategies for conventional software

Test strategies for Conventional Software

There are many strategies that can be used to test software.

At one extreme, you can wait until the system is fully constructed and then conduct tests on the overall system in hopes of finding errors.

This approach simply does not work. It will result in buggy software.

At the other extreme, you could conduct tests on a daily basis, whenever any part of the system is constructed.

This approach, although less appealing to many, can be very effective.

A testing strategy that is chosen by most software teams falls between the two extremes.

It takes an incremental view of testing,

Beginning with the testing of individual program units,

Moving to tests designed to facilitate the integration of the units, Culminating with tests that exercise the constructed system.

Unit Test

Unit testing focuses verification effort on the smallest unit of software design—the software component or module.

The unit test focuses on the internal processing logic and data structures within the boundaries of a component.

This type of testing can be conducted in parallel for multiple components. Unit tests are illustrated schematically in previous Figure.

The module interface is tested to ensure that information properly flows into and out of the program unit under test.

Local data structures are examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution.

All independent paths through the control structure are exercised to ensure that all statements in a module have been executed at least once.

Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.

Finally, all error-handling paths are tested

Good design anticipates error conditions and establishes error-handling paths to reroute or cleanly terminate processing when an error does occur.

Unit testing is simplified when a component with high cohesion is designed. Integration Testing

Integration testing is a systematic technique for constructing the software architecture while at the same time conducting tests to uncover errors associated with interfacing.

Different Integration Testing Strategies

--Top-down testing

-Bottom-up testing -

-Regression Testing -

-SmokeTesting

Top-down testing

Top-down integration testing is an incremental approach to construction of the software architecture.

Modules are integrated by moving downward through the control hierarchy, beginning with the main control module (main program).

Modules subordinate (and ultimately subordinate) to the main control module are incorporated intothe structure in either a depth-first or breadth-first manner.

Top down testing

Integration Testing

The integration process is performed in a series of five steps

1. The main control module is used as a test driver and stubs are substituted for all components directly subordinate to the main control module.

2. Depending on the integration approach selected (i.e., depth or breadth first), subordinate stubs are replaced one at a time with actual components.

3. Tests are conducted as each component is integrated.

4. On completion of each set of tests, another stub is replaced with the real component.

5. Regression testing (discussed later in this section) may be conducted to ensure that new errors have not been introduced.

The process continues from step 2 until the entire program structure is built. Problem encountered during top-down integration testing.

Top-down strategy sounds relatively uncomplicated, but in practice, logistical problems can arise.

The most common of these problems occurs when processing at low levels in the hierarchy is required to adequately test upper levels.

Stubs replace low-level modules at the beginning of top-down testing; therefore, no significant data can flow upward in the program structure.

As a tester, you are left with three choices:

(1) delay many tests until stubs are replaced with actual modules,

(2) develop stubs that perform limited functions that simulate the actual module, or

(3) integrate the software from the bottom of the hierarchy upward.

Bottom-up Integration Testing

Bottom-up integration testing, It begins construction and testing with atomic modules (i.e., components at the lowest levels in the program structure).

Because components are integrated from the bottom up, the functionality provided by components subordinate to a given level is always available and the need for stubs is eliminated.

A bottom-up integration strategy may be implemented with the following steps...

1. Low-level components are combined into clusters (sometimes called builds) that perform a specific software subfunction.
2. A driver (a control program for testing) is written to coordinate test case input and output.
3. The cluster is tested.
4. Drivers are removed and clusters are combined moving upward in the program structure.

Regression Testing

Regression testing is the re-execution of some subset of tests that have already been conducted to ensure that changes have not propagated unintended side effects.

Whenever software is corrected, some aspect of the software configuration (the program, its documentation, or the data that support it) is changed.

Regression testing helps to ensure that changes (due to testing or for other reasons) do not introduce unintended behavior or additional errors.

Regression testing may be conducted manually, by re-executing a subset of all test cases or using automated capture/playback tools.

It is impractical and inefficient to re execute every test for every program function once a change has occurred....

Regression testing is a type of software testing that seeks to uncover new software bugs, OR

Regression testing is the process of testing, changes to computer programs to make sure that the older programming still works with the new changes. Here changes such as enhancements, patches or configuration changes, have been made to them.

Smoke Testing

Smoke testing is an integration testing approach that is commonly used when product software is developed

Smoke testing is performed by developers before releasing the build to the testing team and after releasing the build to the testing team it is performed by testers whether to accept the build for further testing or not.

It is designed as a pacing (Speedy) mechanism for time-critical projects, allowing the software team to assess the project on a frequent basis.

Smoke Testing Example

For Example in a project there are five modules like login, view user, user detail page, new user creation, and task creation etc. So in this five modules first of all developer perform the smoke testing by executing all the major functionality of modules like user is able to login or not with valid login credentials and after login new user can created or not, and user that is created viewed or not. So it is obvious that this is the smoke testing always done by the developing team before submitting (releasing) the build to the testing team.

Now once the build is released to the testing team than the testing team has to check whether to accept or reject the build by testing the major functionality of the build. So this is the smoke test done by testers

Smoke testing provides a number of benefits when it is applied on complex, time critical software projects.

Integration risk is minimized. Because smoke tests are conducted daily, incompatibilities and other show-stopper errors are uncovered early.

The quality of the end product is improved. Because the approach is construction (integration) oriented, smoke testing is likely to uncover functional errors as well as architectural and component-level design errors. If these errors are corrected early, better product quality will result. Error diagnosis and correction are simplified. Progress is easier to assess

3 a) what is significance formal technical review And explain?
b) Differentiate between proactive and reactive strategies?

a) A formal technical review is a software quality assurance activity performed by software engineers (and others).

The objectives of the FTR 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;
(5) to make projects more manageable.
FTR consist of the following phases:

- Review meeting
- Review Reporting and Record Keeping
- Review Guidelines

The FTR is actually a class of reviews that includes walkthroughs, inspections, round-robin reviews and other small group technical assessments of software.
FTR can be conducting using
-Stakeholders meet
-use cases
-Scenarios
-Interviews ,etc.
-Open interviews
-Closed interviews

b)Risk Strategies

Reactive Strategies
-Very common, also known as fire fighting
-Project team sets resources aside to deal with problems
-Team does nothing until a risk becomes a problem

Proactive Strategies

-risk management begins long before technical work starts, risks are identified and prioritized by importance

-team builds a plan to avoid risk if they can or to minimize risks if they turn into problems.

4.Write a detail note on ISO 9000 quality standards.

Ans: ISO 9000 quality standards:

ISO 9000 describes the elements of a quality assurance system in general terms. These elements include the organizational structure, procedures, processes, and resources needed to implement quality planning, quality control, quality assurance and quality improvement.

ISO 9001 is the quality assurance standard that applies to software engineering

The requirements delineated by ISO 9001 address topics such as management responsibility, quality system, contract review, design control, document and data control, product identification and traceability, process control, inspection and testing, corrective and preventive action, control of quality records, internal quality audits, training, servicing, and statistical techniques.

5.Demonstrate risk identification. Describe developing a risk table?

Risk identification is a systematic attempt to specify threats to the project plan. There are two distinct types of risks.

1) Generic risks and
2) product-specific risks.

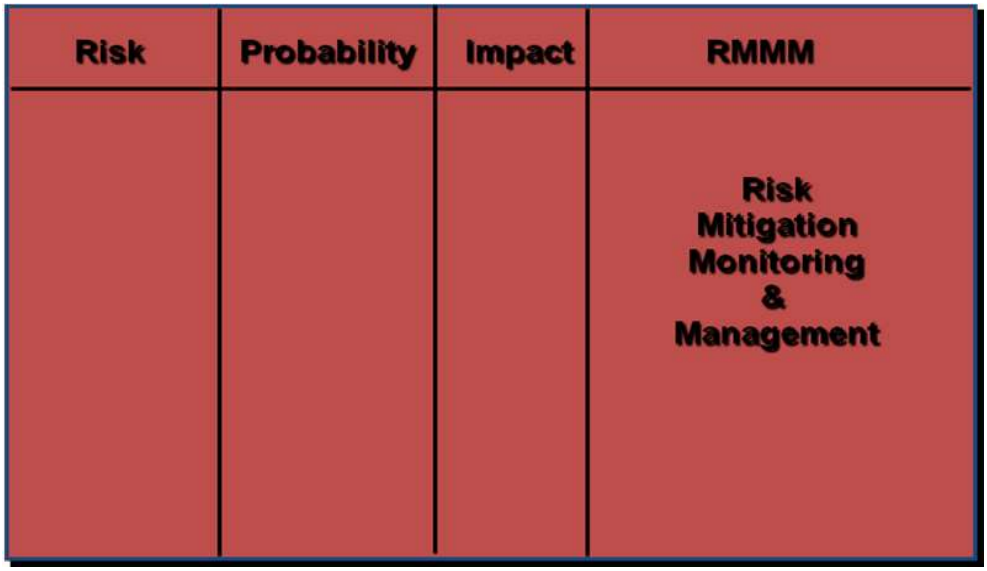
Generic risks are a potential threat to every software project.

Product-specific risks can be identified only by those with a clear understanding of the technology, the people, and the environment that is specific to the project that is to be built.

Known and predictable risks in the following generic subcategories:

- ***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.

Risk table:



- A project team begins by listing all risks (no matter how remote) in the first column of the table.
- Each risk is categorized in Next; the impact of each risk is assessed.
- The categories for each of the four risk components—performance, support, cost, and schedule— are averaged to determine an overall impact value.
- High-probability, high-impact risks percolate to the top of the table, and low-probability risks drop to the bottom. This accomplishes first-order risk prioritization.

The project manager studies the resultant sorted table and defines a cutoff line.
The ***cutoff line*** (drawn horizontally at some point in the table) implies that only risks that lie above the line will be given further attention. Risks that fall below the line are re-evaluated to accomplish second-order prioritization.

Assessing Risk Impact

Three factors affect the consequences that are likely if a risk does occur: its nature, its scope, and its timing.

- ✓ The ***nature*** of the risk indicates the problems that are likely if it occurs.
- ✓ The ***scope*** of a risk combines the severity (just how serious is it?) with its overall distribution.
- ✓ Finally, the ***timing*** of a risk considers when and for how long the impact will be felt.

The overall ***risk exposure***, RE, is determined using the following relationship RE = *P* x *C*

Where *P* is the probability of occurrence for a risk, and *C* is the cost to the project should

the risk occur.

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.

Risk exposure. *RE* = 0.80 x 25,200 ~ $20,200.

The total risk exposure for all risks (above the cutoff in the risk table) can provide a means for adjusting the final cost estimate for a project etc.

6. Demonstrate what type of risks occurs during software development process?

Building and maintaining software can be a risky business. Most enterprises rely on software – so extra cost, delays, or the inability to realise goals can have serious consequences. Larger risks that can sabotage long-term projects require immediate attention. And that means putting the emphasis on risk management. Here, we'll elaborate the top ten risks involved in software development.

1. Estimation and scheduling

The unique nature of individual software projects creates problems for developers and managers in estimating and scheduling development time. Always monitor existing projects so that you apply lessons learnt in the future.

2. Sudden growth in requirements

As a project progresses, issues that are not identified earlier can create a last-minute hurdle to meeting deadlines. Try to think big early on in the project, and anticipate the worst-case or heaviest-use scenario.

3. Employee turnover

Every project has a number of developers working on it. When a developers leaves, he or she may take critical information with him/her. This can delay, and sometimes derail an entire project. Ensure you have resources where team members can collaborate and share knowledge.

4. Breakdown of specification

During the initial phases of integration and coding, requirements might conflict. Moreover, developers may find that even the specification is unclear or incomplete.

5. Productivity issues

On projects involving long timelines, developers tend to take things easy to begin with. As a result, sometimes, they lose significant time to complete the project. Set a realistic schedule, and stick to it.

6. Compromising on designs

In order to get stuck into the next 'real' tasks, developers tend to rush the design-process. This is a waste of programming hours, as designing is the most critical part of software development.

7. Gold plating

Developers sometimes like to show off their skills by adding unnecessary features. For instance, a developer might add Flash to a basic login module to make it look 'stylish'. Again, this is a waste of programming hours.

8. Procedural risks

Day-to-day operational activities might hamper due to improper process implementation, conflicting priorities, or a lack of clarity in responsibilities.

9. Technical risks

Sometimes software development firms reduce the functionality of the software to compensate for overruns pertaining to high budgets and scheduling. There is always a conflict between achieving maximum functionality of the software and peak performance. In order to compensate for excessive budget and schedule overruns, companies sometimes reduce the functionality of the software.

10. Unavoidable risks

These include changes in government policy, the obsolescence of software or other risks that cannot be controlled or estimated. As the field of software development becomes more and more complex, the risks associated with it have intensified. It is vital that development firms focus on strategic planning to mitigate such risks.

7.Explain sprinciples of risk management.

Risk Management is an approach that helps in managing and make best use of the available resources. A computer code project may be laid low with an out sized 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 is 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.

Principles of Risk Management:

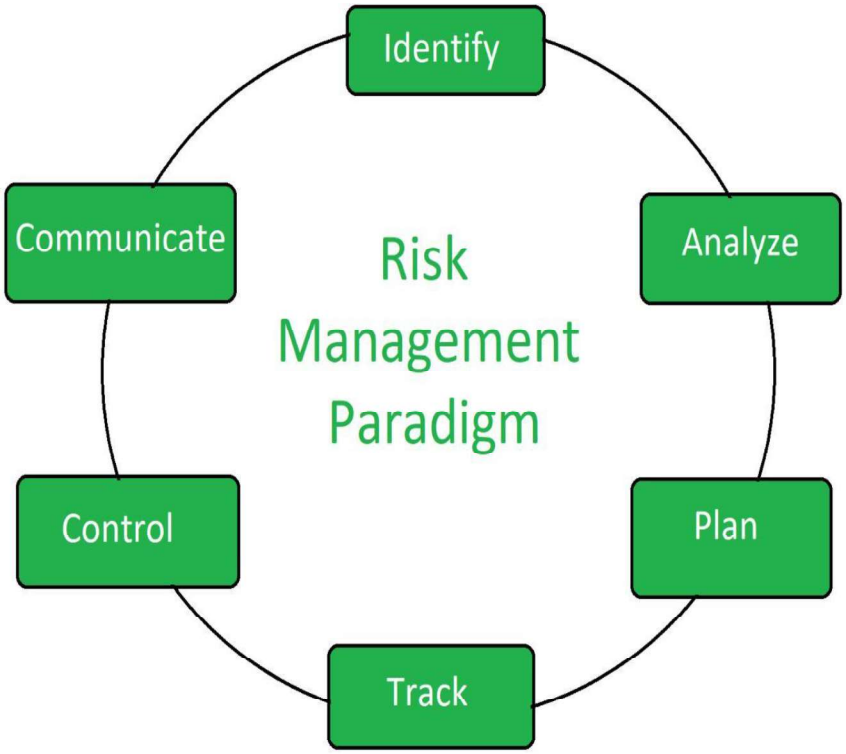
There are 5 principles of Risk Management. They are:

1. **Global Perspective:**
   Larger system definitions, design and implementation is considered. The opportunity and the impact that the risk is going to have is looked. View software risks in the context of a system and the business problem planned to solve.

1. **orward Looking View:**
   Looking at the possible uncertainties that might dragged. Possible solutions of the risks that might occur in the future are considered. Think about the risk which may occur in the future and create future plans for managing the future events.
2. **Open Communication:**
   This enables the free flow of the communication between the end users and the development team so that they can clarify the risks. Encourage all the stakeholders and users for suggesting risks at any time.
3. **Integrated Management:**
   Risk management is made an integral part of the project management during this phase. A consideration of risk should be integrated into the software process.
4. **Continuous Process:**
   Risks are tracked continuously throughout the risk management paradigm during this phase. Modify the identified risk than the more information is known and add new risks as better insight is achieved.

Risk Management Paradigm:

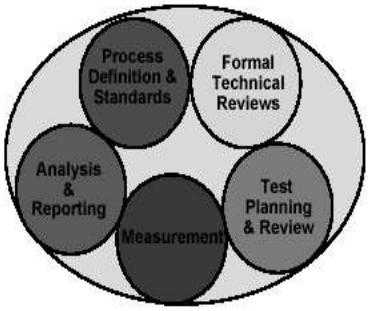
1. **Identify:**
   Risks are identified before major problem is created. If the risks are identified before they create a major problem then there might not be more difficulty in controlling the risks.
2. **Analyze:**
   Deep analysis of nature, behavior and type of risk and collect information about it. It is required for the purpose of the determination of the knowledge about the risk.
3. **Plan:**
   Convert the plan into actions and implementation. This phase includes the actions and implementation of the planning that was done before. After the risk detection plans are made and executed.
4. **Track:**
   Necessary actions are monitored. Necessary action means the required work for the removal and minimization of the risk detected.



1. **Identify:**
   Risks are identified before major problem is created. If the risks are identified before they create a major problem then there might not be more difficulty in controlling the risks.
2. **Analyze:**
   Deep analysis of nature, behavior and type of risk and collect information about it. It is required for the purpose of the determination of the knowledge about the risk.
3. **Plan:**
   Convert the plan into actions and implementation. This phase includes the actions and implementation of the planning that was done before. After the risk detection plans are made and executed.
4. **Track:**
   Necessary actions are monitored. Necessary action means the required work for the removal and minimization of the risk detected.
5. **Control:**
   Correct the deviation and make necessary changes. Put the right thing in the right place and the required field will changed according to the changes required.
6. **Communicate:**
   Discussion about the current risks and the future risks and their management. Make a productive discussion between the developer and tester on the risks found in the software.

8. Explain software quality assurance. Explain risk projection in detail.

SOFTWARE QUALITY ASSURANCE



•Software Quality can be defined as Conformance to explicitly state functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.

•Definition serves to emphasize three important points:

•Software requirements are the foundation from which quality is measured. Lack of conformance to requirements is lack of quality.

•Specified standard define a set of development criteria that guide the manner in which software is engineered. If the criteria are not followed, lack of quality will almost surely result.

•A set of implicit requirements often goes unmentioned. If software conforms to its explicit requirements but fails to meet implicit requirements, software quality is suspect.

SQA Activities

•Prepares an SQA plan for a project.

•The plan identifies

•Evaluations to be performed.

•Audits and reviews to be performed.

•Standards that is applicable to the project.

•Procedures for error reporting and tracking.

•Documents to be produced by the SQA group.

•Amount of feedback provided to the software project team.

•Participates in the development of the project's software process description.

• 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.

•Reviews software engineering activities to verify compliance with the defined software process.

•Identifies, documents, and tracks deviations from the process and verifies that corrections have been made.

•Audits designated software work products to verify compliance with those defined as part of the software process.

•Reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made

•Periodically reports the results of its work to the project manager.

•Ensures that deviations in software work and work products are documented and handled according to a documented procedure.

•Records any noncompliance and reports to senior management.

•Noncompliance items are tracked until they are resolved.

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

•The consequences of the problems associated with the risk, should it occur.

• There are four risk projection steps:

•establish a scale that reflects the perceived likelihood of a risk

•delineate the consequences of the risk

•estimate the impact of the risk on the project and the product,

•note the overall accuracy of the risk projection so that there will be no misunderstandings.

Developing a Risk Table

Risk	Probability	Impact	RMMM
			Risk Mitigation
			Monitoring &
			Management

•Estimate the probability of occurrence

•Estimate the impact on the project on a scale of 1 to 5, where

• 1 = low impact on project success

• 5 = catastrophic impact on project success

•sort the table by probability and impact

Assessing Risk Impact

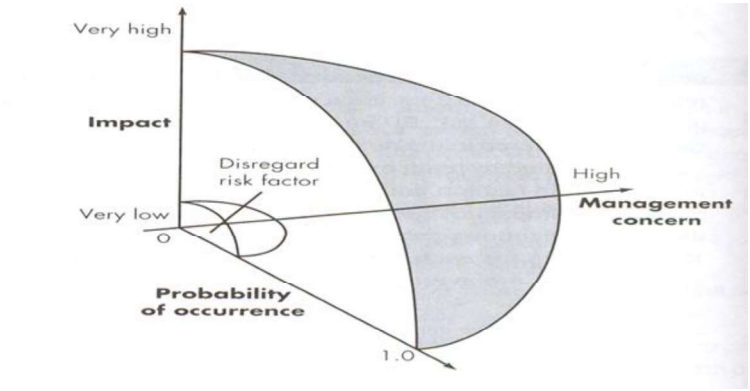
- The overall risk exposure, RE, is determined using the following relationship:

RE = P x C

where

P is the probability of occurrence for a risk, and

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



9.Define software reliability along with its terms?

Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. Software Reliability is also an important factor affecting system reliability. It differs from hardware reliability in that it reflects the design perfection, rather than manufacturing perfection. The high complexity of software is the major contributing factor of Software Reliability problems.

Software Reliability is not a function of time - although researchers have come up with models relating the two. The modeling technique for Software Reliability is reaching its prosperity, but before using the technique, we must carefully select the appropriate model that can best suit our case. Measurement in software is still in its infancy. No good quantitative methods have been developed to represent Software Reliability without excessive limitations. Various approaches can be used to improve the reliability of software, however, it is hard to balance development time and budget with software reliability.

A good software reliability engineering program, introduced early in the development cycle, will mitigate these problems by: Preparing program management in advance for the testing effort and allowing them to plan both schedule and budget to cover the required testing.

Continuous review of requirements throughout the life cycle, particularly for handling of exception conditions. If requirements are incomplete there will be no testing of the exception conditions.

SoHaR software reliability engineers are experienced in all the stages and tasks required in a comprehensive software reliability program. We can support or lead tasks such as:

1) Reliability Allocation

2) Defining and Analyzing Operational Profiles

3) Test Preparation and Plan

4) Software Reliability Models

Reliability Allocation:-

Reliability allocation is the task of defining the necessary reliability of a software item. The item may be part of an integrated hardware/software system, may be a relatively independent software

application, or, more and more rarely, a standalone software program. In either of these cases our goal is to bring system reliability within either a strict constraint required by a customer or an internally perceived readiness level, or optimize reliability within schedule and cost constraints.

SoHaR will assist your organization in the following tasks:

Derive software reliability requirements from overall system reliability requirements.

When possible, depending on lifecycle stage and historical data, estimate schedule and cost dependence on software reliability goals.

Optimize reliability/schedule/cost based on your constraints and your customer's requirements,

Defining and Analyzing Operational Profiles:-

The reliability of software, much more so than the reliability of hardware, is strongly tied to the operational usage of an application. A software fault may lead to system failure only if that fault is encountered during operational usage. If a fault is not accessed in a specific operational mode, it will not cause failures at all. It will cause failure more often if it is located in code that is part of an often used "operation" (An operation is defined as a major logical task, usually repeated multiple times within an hour of application usage). Therefore in software reliability engineering we focus on the operational profile of the software which weighs the occurrence probabilities of each operation. Unless safety requirements indicate a modification of this approach we will prioritize our testing according to this profile.

SoHaR will work with your system and software engineers to complete the following tasks required to generate a useable operational profile:

Determine the operational modes (high traffic, low traffic, high maintenance, remote use, local use etc).

Determine operation initiators (components that initiate the operations in the system).

Determine and group "Operations" so that the list includes only operations that are significantly different from each other (and therefore may present different faults).

Determine occurrence rates for the different operations.

Construct the operational profile based on the individual operation probabilities of occurrence.

Test Preparation and Plan:-

Test preparation is a crucial step in the implementation of an effective software reliability program. A test plan that is based on the operational profile on the one hand, and subject to the reliability allocation constraints on the other, will be effective at bringing the program to its reliability goals in the least amount of time and cost.

Software Reliability Engineering is concerned not only with feature and regression test, but also with load test and performance test. All these should be planned based on the activities outlined above.

The reliability program will inform and often determine the following test preparation activities:

Assessing the number of new test cases required for the current release.

New test case allocation among the systems (if multi-system).

New test case allocation for each system among its new operations.

Specifying new test cases

Adding the new test cases to the test cases from previous releases.

Software Reliability Models:-

Software reliability engineering is often identified with reliability models, in particular reliability growth models. These, when applied correctly, are successful at providing guidance to management decisions such as:

Test schedule

Test resource allocation

Time to market

Maintenance resource allocation

10.List the seven principals of risk management and explain?

Risk management is the evaluation, prioritization, and identification of risks. This is then coupled with the economic application of resources to monitor the impact of unfortunate events. Many businesses employ trained personnel to help deal with real risks. Some organizations go to the extent of setting apart departments to specialize in the management of risks.
The following risk management steps and principles are essential in the success of any business:

Step 1: Establish the Context

The content defines the scope of risk management. Risks may arise from external or internal factors (external or internal context). The external context is the environment in which a business operates. In the SWOT analysis, the external context looks into the opportunities and threats. The internal context is the internal environment of a company. Factors to look into the internal context include strengths, weaknesses, structure, roles, governance, and accountability.

Step 2: Identification of Risk

Risk identification lays the foundation for risk management. It involves identifying potential risks that could hinder the business from achieving set objectives and requires knowledge of the social, political, economic environment and market in which the company operates. Poor risk identification may result in significant losses. The impact of a risk on business can only be controlled if the risk is identified beforehand.

Step 3: Communicating the Risk

For effective communication, there has to be a communicator and an audience. The communicator's primary objective is to make known the potential hazards and risks to all the internal publics of the organization. Effective communication of risk helps in diffusing problems, ranging from damage of property to litigation and losses.

Step 4: Analyzing Risk

In risk analysis, you first need to identify the potential threats and then establish what the probability is of the risk materializing. An effective risk analysis should make it easy to understand the primary risks. If the analysis is carried out effectively, it aids in a business's decision-making mechanism. The fundamental impediment in risk analysis is determining the rate of occurrence since not all past events provide statistical information.

Step 5: Prioritize the Risk

Having an extensive list of risks is unsettling. You can, however, manage this by simply categorizing risks according to magnitude. The risk could be high, medium or low. The high risks can derail the progress of a project and hence need immediate attention. This is not to mean the other risks are not significant, but they are probably not serious enough to threaten the success of your project. Risk prioritization enhances efficiency in the management of risks.

Step 6: Potential Risk Treatment

The techniques to manage risk are classified into four major categories.

a. Risk transfer – The expected party transfers losses to another party for costs, especially in the case of insurance contracts.

b. Risk avoidance – This is when you completely avoid activities that could result in risks.

c. Risk retention – Two main ways of achieving this is by self-insurance and captive insurance.

d. Risk control – This can be done by either avoidance or risk control.

7. Come Up with a Risk Plan

The risk plan should propose working security controls for managing risks. An effective risk management plan involves making a list, prioritizing risks, developing an action plan, human resource deployment and communication.

Step 7: Implementation

For risks that are to be transferred, purchase insurance policies. Keep off all risks that can be avoided and follow the established methods of dealing with risks.

Management of risks is critical to any organization. Managers must develop a holistic approach to running their departments. There is no 'one size fits all' when it comes to managing risks. Risks are specific to businesses, and what is risky for one business may be a non-issue for another.