UNIT V Risk Management and Quality Management

Risk Management

Risk Management:

Risk is an undesired event or circumstance that may occur during a project. Risk management involves anticipating, identifying, analyzing, and managing various risks that a project may be susceptible to. There are reactive and proactive risk management strategies.

Reactive Vs Proactive Risk Strategies

- **Reactive:**
  - Monitors likely risks during the project, and resources are set aside to address them as they arise.
- **Proactive:**
  - Risks are identified in advance, and their probability and impact are assessed.
  - Aims to minimize the impact of risks before they occur.

Software Risks

Software risks involve two characteristics:

1. **Uncertainty:**
   - Risks may or may not happen.
2. **Loss:**
   - If a risk becomes a reality, unwanted loss or consequences will occur.

Categories of Software Risks:

1. **Project Risk:**
   - Threatens the project plan and affects the schedule and resultant cost.
2. **Technical Risk:**
   - Threatens the quality and timeliness of the software to be produced.
3. **Business Risk:**
   - Threatens the viability of the software to be built.
4. **Known Risk:**

- Risks that can be identified through careful evaluation.

5. **Predictable Risk:**
   - Risks identified based on past project experience.
6. **Unpredictable Risk:**
   - Risks that occur and may be difficult to identify in advance.

Risk Management Process:

1. **Risk Identification:**
   - Identifying potential risks that could impact the project.
2. **Risk Projection:**
   - Assessing the probability and impact of identified risks.
3. **Risk Refinement:**
   - Detailed analysis and further understanding of the risks.
4. **Risk Mitigation, Monitoring, and Management (RMMM):**
   - Developing a plan to handle and mitigate risks.
5. **RMMM Plan:**
   - Documented plan outlining strategies for risk management.

Risk Identification

Risk identification is a crucial step in the risk management process. It involves identifying all possible risks, creating item checklists, categorizing risks into components (such as performance risk, cost risk, support risk, and schedule risk), and assessing the severity of risks.

Steps in Risk Identification:

1. **Identify All Possible Risks:**
   - Comprehensive identification of potential risks that could impact the project.
2. **Create Item Checklists:**
   - Developing checklists to systematically identify and document potential risks.
3. **Categorize into Risk Components:**
   - Grouping identified risks into categories based on their nature (e.g., performance, cost, support, schedule).
4. **Risk Severity Assessment:**
   - Dividing risks into categories based on their severity:

- Negligible (0)
- Marginal (1)
- Critical (2)

Risk Identification Components:

Risk identification involves considering various components related to the project and its environment:

- Product Size
- Business Impact
- Development Environment
- Process Definition
- Customer Characteristics
- Technology to be Built
- Staff Size and Experience

Risk Projection

Risk projection, also known as risk estimation, estimates the impact of identified risks on the project and the product. This estimation is typically done using a Risk Table, which categorizes risks based on their likelihood and consequences.

Steps in Risk Projection:

1. Estimate Likelihood (Li) for Each Risk:
   - Assess the probability of each identified risk occurring.
2. Estimate Consequences (Xi):
   - Determine the potential consequences or impact of each risk.
3. Estimate Impact:
   - Evaluate the overall impact of each risk.
4. Draw the Risk Table:
   - Categorize risks based on their likelihood and consequences.

Consideration in Risk Projection:

- Ignore risks with low management concern (low impact or low probability).
- Consider all risks with high management concern (high impact or moderate/high probability).

Risk Impact Categories:

The impact of each risk is assessed using impact values:

- Catastrophic (1)
- Critical (2)
- Marginal (3)
- Negligible (4)

Risk Refinement

Risk refinement, also known as risk assessment, involves reviewing the risk impact and refining the risk table based on the nature, scope, and timing of potential problems. It is a crucial step in the risk management process.

Factors for Risk Refinement:

1. Nature:
   - Examining likely problems that may occur if the risk materializes.
2. Scope:
   - Assessing the seriousness of the risk.
3. Timing:
   - Determining when the risk may occur and how long it may impact the project.

Risk Elaboration:

Risk refinement is based on risk elaboration, which involves providing detailed information about each identified risk, including its potential impact and consequences.

Risk Exposure Calculation:

Risk exposure (RE) is calculated using the formula:
$RE = P \times C$

Where:

- (P) is the probability of the risk occurring.
- (C) is the cost of the project if the risk materializes.

RMMM

Risk Mitigation, Monitoring, and Management

The goal of RMMM is to assist the project team in developing a comprehensive strategy for dealing with risks. It addresses three key issues:

1. Risk Avoidance:
    - Proactive planning to avoid potential risks.
2. Risk Monitoring:
    - Assessing whether predicted risks occur.
    - Ensuring that risk aversion steps are properly applied.
    - Collecting information for future risk analysis.
3. Risk Management:
    - Contingency planning for actions to be taken in case mitigation steps fail and the risk becomes a live problem.

RMMM Plan:

The RMMM plan documents all work performed as part of risk analysis. It includes:

- Risk Information Sheet (RIS):
    - Documenting each risk individually.
    - Maintained using a database system.
- Risk Mitigation Monitoring And Management Plan (RMMP):
    - Outlining actions to be taken in the event that mitigation steps fail.

The RMMM plan provides a structured approach to handling risks throughout the project lifecycle.

Quality Management

Quality management ensures that the software development process adheres to high-quality standards. It involves various aspects such as quality concepts, software quality assurance, software reviews, statistical software quality assurance, the Capability Maturity Model Integration (CMMI), software reliability, and the ISO 9000 quality standards.

- Quality Concepts:
    - Fundamental principles of quality in software development.
- Software Quality Assurance (SQA):
    - The process of monitoring and improving the software engineering process to ensure quality.
- Software Reviews:

    - Formal assessments of software work products.
- **Statistical Software Quality Assurance:**
    - The use of statistical techniques for quality assurance.
- **Capability Maturity Model Integration (CMMI):**
    - A model that provides guidance for process improvement across various aspects of an organization.
- **Software Reliability:**
    - The probability of failure-free software operation for a specified period in a specified environment.
- **ISO 9000 Quality Standards:**
    - A set of international standards for quality management and quality assurance.

Quality Concepts

1. **Variation Control:**
    - Variation control is the core of quality control.
    - Aims to minimize the difference between predicted and actual resources used in various projects, including staff, equipment, and calendar time.
2. **Quality of Design:**
    - Refers to the characteristics specified by designers for the end product.
    - Focuses on defining the desired features and attributes of the final product.

Quality Management:

1. **Quality of Conformance:**
    - Reflects the degree to which design specifications are followed during the manufacturing of the product.
2. **Quality Control:**
    - Involves a series of inspections, reviews, and tests used to ensure that a work product conforms to its specifications.
3. **Quality Assurance:**
    - Encompasses auditing and reporting functions to assess the effectiveness and completeness of quality control activities.

Cost of Quality:

1. **Prevention Costs:**

- Include activities such as quality planning, formal technical reviews, test equipment, and training to prevent defects.

2. **Appraisal Costs:**
   - Encompass in-process and inter-process inspection, equipment calibration and maintenance, and testing activities.
3. **Failure Costs:**
   - Include costs related to rework, repair, and failure mode analysis.
4. **External Failure Costs:**
   - Encompass complaint resolution, product return and replacement, help line support, and warranty work.

Software Quality Assurance (SQA)

1. **Conformance to Software Requirements:**
   - The foundation from which software quality is measured.
   - Ensures that the software meets the specified requirements.
2. **Use of Specified Standards:**
   - Specified standards are employed to define the development criteria that guide the software engineering process.
   - Standards provide a framework for consistent and effective software development.
3. **Conformance to Implicit Requirements:**
   - Software must adhere to both explicit requirements (specified in documentation) and implicit requirements (e.g., ease of use, maintainability, reliability).

SQA Activities:

1. **Prepare SQA Plan:**
   - Develop a Software Quality Assurance Plan for the project.
   - Outlines the approach and activities related to ensuring software quality.
2. **Participate in Process Description:**
   - Contribute to the development of the project's software process description.
   - Collaborate in defining the processes that guide software development.
3. **Review Software Engineering Activities:**
   - Conduct reviews of software engineering activities to verify compliance with the defined software process.

- Ensure that activities align with established processes and standards.

4. **Audit Software Work Products:**
   - Perform audits on designated software work products to verify compliance with the defined software process.
   - Ensure that the work products meet the specified standards.
5. **Handle Deviations:**
   - Ensure that any deviations in software or work products are documented.
   - Establish procedures for handling and addressing deviations from the defined process.
6. **Record Noncompliance Evidence:**
   - Record evidence of any noncompliance with the established processes.
   - Report instances of noncompliance to management for appropriate action.

Software Reviews

Purpose of Software Reviews:

1. **Find Errors Early:**
   - Identify errors in software artifacts before they are passed to subsequent software engineering activities or released to the customer.
   - A proactive approach to ensure software quality.
2. **Formal Technical Reviews (FTRs):**
   - Conducted by software engineers and other team members.
   - Focus on quality control and improvement of software.
3. **Effectiveness of FTRs:**
   - FTRs, such as walkthroughs or inspections, are effective means for enhancing software quality.
   - Uncover errors in function, logic, or implementation.
   - Verify compliance with requirements and predefined standards.

Formal Technical Reviews

Review Meeting in FTR:

1. **Membership:**
   - Consists of three to five members.

- Each person should prepare for the meeting in less than two hours.

2. **Duration:**
   - Meeting duration should be less than two hours.
3. **Focus:**
   - Centered on a specific work product, such as requirement specifications, detailed component design, or source code listing.
4. **Initiation:**
   - The producer informs the project leader of the completion of the work product and the need for a review.
   - The project leader contacts a review leader, who evaluates product readiness and organizes the review.
5. **Preparation:**
   - The review leader distributes copies of the product material to two or three review members for advance preparation.
   - Reviewers spend one to two hours reviewing the product and making notes.
6. **Meeting Conduct:**
   - Attended by the review leader, all reviewers, and the producer.
   - The producer introduces the product, walks through it, and reviewers raise prepared issues.
   - Errors found are noted by a recorder.

Review Reporting and Record Keeping:

1. **Recording Issues:**
   - A recorder documents all issues raised during the review.
2. **Review Summary Report:**
   - Answers key questions: What was reviewed? Who reviewed it? What were the findings and conclusions?
   - A single-page form with possible attachments.
3. **Review Issues List:**
   - Serves to identify problem areas in the product.
   - Acts as an action item checklist guiding the producer in making corrections.

Review Guidelines:

1. **Focus on the Product:**

- Review the software artifact, not the producer.

2. **Set an Agenda:**
   - Establish and adhere to an agenda for the review.
3. **Limit Debate:**
   - Limit debate and rebuttal during the review.
4. **Enunciate Problem Areas:**
   - Clearly state problem areas but avoid solving every issue during the review.
5. **Take Return Notes:**
   - Record key points and issues discussed during the review.
6. **Limit Participants:**
   - Keep the number of participants limited and ensure advance preparation.
7. **Checklist Development:**
   - Develop a checklist for each product likely to be reviewed.
8. **Allocate Resources:**
   - Allocate resources and schedule time for FTRs.
9. **Conduct Training:**
   - Provide meaningful training for all reviewers.
10. **Review Early Reviews:**
    - Evaluate and improve the effectiveness of early reviews.

Software Defects:

- Design activities contribute 50-65% of all defects during the software process.
- Review techniques have proven to be up to 75% effective in uncovering design flaws, reducing the cost of subsequent activities.

Statistical Software Quality Assurance

Information Collection:

- Collect and categorize information about software defects.
- Trace each defect back to its cause.

Pareto Principle:

- Utilize the Pareto principle, where 80% of defects can be traced to 20% of causes.
- Identify the "vital few" defect causes.

Defect Resolution:

- Isolate and address the root causes of defects in the "vital few."

Six Sigma for Software Engineering:

Core Steps:

1. **Define:**
   - Clearly define customer requirements, deliverables, and project goals through effective communication methods.
2. **Measure:**
   - Measure each existing process and its output to determine current quality performance (e.g., compute defect metrics).
3. **Analyze:**
   - Analyze defect metrics to identify the "vital few" causes.

Improvement for Existing Process:

1. **Improve:**
   - Enhance the existing process by eliminating root causes for defects.
2. **Control:**
   - Implement controls to ensure that future work does not reintroduce causes of defects.

New Process Development:

1. **Design:**
   - Design each new process to avoid root causes of defects and align with customer requirements.
2. **Verify:**
   - Verify that the process model will prevent defects and meet customer requirements.

Key Principles:

1. **Information Traceability:**

- Trace each defect to its origin to understand the causes.

2. **Pareto Principle Application:**
   - Focus on addressing the "vital few" causes that contribute to the majority of defects.
3. **Continuous Improvement:**
   - Emphasize continuous improvement by eliminating root causes and controlling future work.
4. **Customer-Centric Approach:**
   - Define customer requirements and goals to align processes with customer needs.
5. **Data-Driven Decision-Making:**
   - Use defect metrics and data to drive decisions about process improvement.
6. **Prevention in Process Design:**
   - Design new processes to prevent defects by avoiding root causes from the outset.
7. **Verification of Process Models:**
   - Verify that process models align with customer requirements and are defect-resistant.

Software Reliability

Definition:

- Software reliability is defined as the probability of failure-free operation of a computer program in a specified environment for a specified time period.

Measurement and Estimation:

- It can be measured directly and estimated using historical and developmental data.
- Software reliability problems are often traced back to errors in design or implementation.

Measures of Reliability:

1. **Mean Time Between Failure (MTBF):**
   - MTBF = MTTF + MTTR

- MTTF (Mean Time To Failure) is the average time a system runs between failures.
- MTTR (Mean Time To Repair) is the average time it takes to repair a system after a failure.

2. **Availability:**
   - Availability = [(MTTF / (MTTF + MTTR))] x 100%

The ISO 9000 Quality Standards

Definition:

- ISO (International Standards Organization) is a consortium of 63 countries established to plan and foster standardization.
- The ISO 9000 series of standards, declared in 1987, provides guidelines for maintaining a quality system.

Types of ISO 9000 Standards:

1. **ISO 9001:**
   - Applies to organizations engaged in design, development, production, and servicing of goods, including most software development organizations.
2. **ISO 9002:**
   - Applies to organizations involved only in production, not design. Examples include steel and car manufacturing industries.
3. **ISO 9003:**
   - Applies to organizations involved only in the installation and testing of products.

ISO Certification Process:

1. **Application:**
   - The organization applies to the ISO registrar office for registration.
2. **Pre-Assessment:**
   - The registrar conducts a rough assessment of the organization.
3. **Document Review and Adequacy of Audit:**
   - The registrar reviews the organization's documents and suggests improvements.
4. **Compliance Audit:**

- The registrar checks if the organization has implemented the suggested improvements.

5. **Registration:**
   - The ISO certification is awarded by the registrar after successful completion of all stages.
6. **Continued Inspection:**
   - The registrar continues to monitor the organization periodically.

ISO 9000 standards emphasize a quality system approach, ensuring that proper stages are followed for production, leading to high-quality products. Certification is awarded to organizations that successfully implement these quality practices.