Lecture 8 - Software Quality Assurance, Control, and Change Management

---

1. Software Quality Assurance (SQA)

- **Definition:**
  A planned and systematic application of activities to ensure conformance of software processes and products to requirements, standards, and procedures.
- **Key Focus:**
  - **Ensuring all elements of the operational environment conform to quality requirements.**
  - **Activities are planned and systematic.**
- **SQA Process:**
  - **Quality Planning:**
    - **Identifies standards, procedures, and resource allocations for achieving quality.**
    - **Output: A Quality Assurance Planning Document.**
  - **Quality Assurance:**
    - **Ensures compliance with quality standards throughout the project.**
  - **Quality Control:**
    - **Validates that deliverables meet quality standards using inspections, analyses, and testing.**
- **Balancing Constraints:**
  - **Quality must align with other project constraints like time, budget, and criticality (e.g., NASA's software classification system).**

---

2. Quality Control

- **Goal:**
  To ensure deliverables conform to quality standards outlined during planning.
- **Main Tools:**
  1. **Inspections: Manual reviews of artifacts.**
  2. **Analyses: Static and dynamic checks for anomalies.**
  3. **Testing: Evaluating performance under specific conditions.**
- **Challenges:**
  1. **Complexity in assessing non-functional requirements (e.g., maintainability, usability).**
  2. **Cost and difficulty of test automation (e.g., GUI testing).**
  3. **Diverse technologies in modern systems (e.g., HTML, JavaScript, OS).**
- **Techniques:**
  1. **Walkthroughs and Code Inspections: Independent team review.**
  2. **Static Checkers: Verify syntax correctness.**

3. **Dynamic Checkers: Monitor execution for anomalies.**
4. **Formal Verification: Prove system properties using abstract models.**
5. **Code Metrics: Quantitative measures like cyclomatic complexity, inheritance depth, and unit test coverage.**

---

3. Metrics Collection in Quality Management

- **Purpose:**
  - **To quantitatively evaluate how well project goals are being achieved.**
  - **Trends provide better insights than static numbers.**
- **Types of Metrics:**
  - **Process Metrics: Evaluate the project process itself.**
  - **Product Metrics: Evaluate the software product, including:**
    - **Size Metrics: Source lines of code (SLOC), number of classes.**
    - **Complexity Metrics: Cyclomatic complexity, coupling, inheritance depth.**
- **Considerations:**
  - **Automating metrics collection improves efficiency.**
  - **Function-oriented metrics may require specialized expertise.**

---

4. Change Control

- **Definition:**
  A set of practices ensuring that all change requests are managed systematically and effectively.
- **Key Concepts:**
  - **Configuration Management: Maintains consistency in project outputs over time.**
  - **Change Causes:**
    - **Incomplete requirements.**
    - **Better system understanding.**
    - **Technical opportunities or challenges.**
    - **External changes (e.g., market trends).**
- **Process:**
  - **A Change Control Board (CCB) may oversee and approve/reject changes.**
  - **Costs and risks of changes grow as the project progresses.**
  - **Agile methodologies treat changes as evolving requirements.**

---

5. Change Management in Practice

- **Challenges:**
  - **Rapid changes due to software's editable nature (e.g., file updates).**
  - **Requires integration with bug reporting and lifecycle management.**
- **Best Practices:**
  - **Maintain formal records of all changes.**
  - **Evaluate the impact of each change systematically.**
  - **Embrace change processes for adaptability in modern development environments.**

---

6. Post-Implementation Considerations

- **Post-Mortem Analysis:**
  - **Critical to learn from successes and failures.**
  - **Structured to include:**
    - **Project Description: Context and background.**
    - **The Good: What worked well.**
    - **The Bad: Key challenges and setbacks.**
    - **The Ugly: Prescriptions for future improvements.**
- **Releasing Staff:**
  - **Transitioning to new activities should acknowledge contributions and allocate meaningful roles.**

---

1. Version Control Systems (VCS)

Main Concepts:

- **Early VCS:**
  - **Each file had an independent repository.**
  - **Coherence across files was managed by assigning the same tags to all artifacts forming a baseline.**
- **Modern VCS:**
  - **Manage sets of artifacts in an integrated manner.**
  - **Support parallel access and editing to accommodate collaborative environments.**
  - **Use tagging to mark important baseline records (e.g., major releases or milestones).**

Key Features of VCS:

1. **Parallel Development:**
   - **Multiple users can access and modify files simultaneously.**

2. **Tagging:**
    - **Allows marking of key snapshots, aiding in tracking and retrieval of stable versions.**

---

2. Risk Management in Software Projects

Motivations:

- **Financial data alone cannot determine project viability.**
- **Planning involves dealing with uncertainties (e.g., time estimation, resource allocation).**
- **Projects operate in non-nominal conditions; unplanned changes are inevitable.**

Definition of Risk:

1. **Traditional View:**
   Risk is the possibility of suffering a loss.
2. **Project Management View:**
    - **Risk refers to events or conditions that can have a positive or negative impact on objectives.**
        - **Negative outcome: Menace.**
        - **Positive outcome: Opportunity.**

Goals of Risk Management:

- **Assess whether a project is worth undertaking.**
- **Refine budgets and schedules for realistic planning.**
- **Enhance the likelihood of project success by staying:**
    - **Within scope.**
    - **Within budget.**
    - **Within quality standards.**
    - **On time.**

Risk Management Objectives:

1. **Increase probability and impact of positive events.**
2. **Decrease probability and impact of negative events.**

Fields Utilizing Risk Management:

- **Finance: Portfolio risk evaluation.**
- **Insurance: Calculating premiums and liabilities.**
- **Engineering: Safety-critical systems and security.**
- **Software Development: Identifying risks during the lifecycle process.**

Relevant Standards:

- **ISO/IEC 12207: Software life cycle processes.**
- **UNI EN 29000-3: Guidelines for applying ISO 9001 to software development.**
- **UNI ISO 10006: Project management guidelines.**

Techniques in Risk Management:

1. **FMEA (Failure Modes and Effects Analysis):**
    - **Identifies potential failure points and their impacts.**
2. **FTA (Fault Tree Analysis):**
    - **Analyzes the root causes of failures systematically.**
3. **Simulation Models:**
    - **Simulates various scenarios to assess potential risks.**