GYAAN SANDHI
YOUR BRIDGE TO BRILLIANCE

UNIT 4

Software Quality Asssurance & Quality Control

Quality & Product Relationship

What is Quality?

- Quality means how well a product or service meets customer requirements.
- High-quality products have fewer defects and better reliability.

What is Productivity?

- Productivity is output produced per unit of input (time, resources, effort).
- High productivity means more work done efficiently.

Relationship Between Quality and Productivity

1. **High Quality Improves Productivity**
    - Fewer defects → less rework → saves time and resources.
    - Well-defined processes → smooth workflow → faster output.

Example:

- Bug-free software reduces time spent on fixing errors, increasing development efficiency.

Low Quality Reduces Productivity

- Defects and errors → rework, delays, and wastage → lower productivity.
- Unclear requirements or poor processes slow down work.

Example:

- Frequent software crashes cause developers to spend extra time debugging instead of adding new features.

Continuous Quality Improvement Boosts Productivity

- Process improvements and preventive actions reduce future errors.
- Teams work more efficiently over time.

Example:

- Automated testing reduces manual effort and increases release speed.

Requirements Of Products

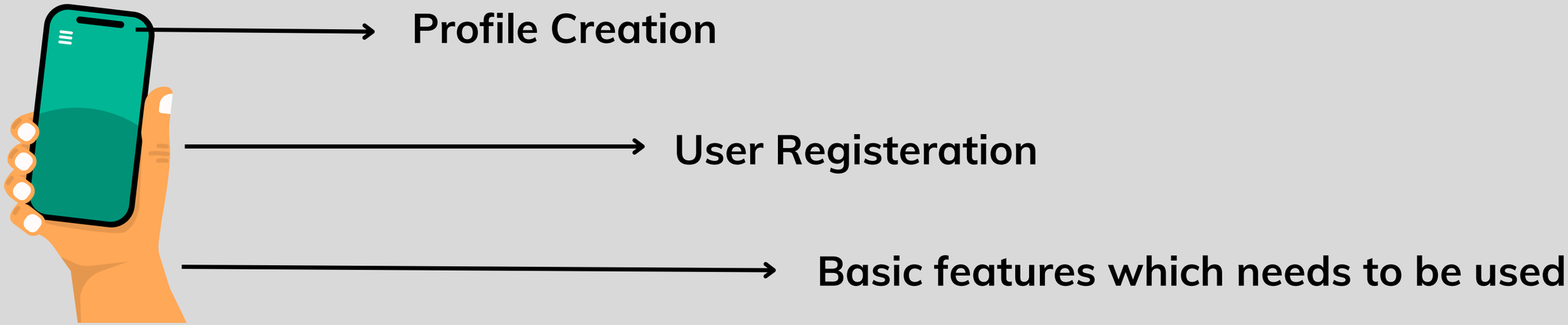
1.Stated Requirement

Requirement

Eg. Application should have login feature with 2 factor authentication

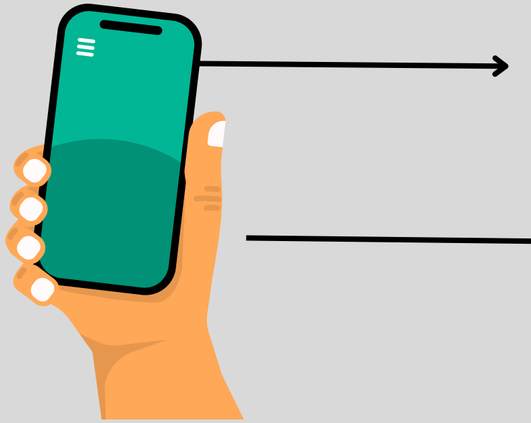
2. Present Requirement

Present time means current needs which is necessary to make the product work better



2(B). Future Requirement

- **These are the needs which are not necessary in present .**
- **But it is needed in future to grow the product.**



Multilingual support

AI based Recommendation

Characteristics of software

1. Functionality

- Functionality shows whether the software performs its main tasks correctly.
- It checks if user requirements are properly fulfilled.

Example:

- A banking app allows money transfer, balance checking, and viewing transaction history.

2. Reliability

- Reliability means the software works consistently without crashing.
- It should perform well even during continuous usage.

Example:

- An airline booking system shows real-time seat availability to avoid double booking.

3. Usability

- Usability focuses on how easy and user-friendly the software is.
- Users should not need special training to use it.

Example:

- A video conferencing app makes scheduling, joining, and screen sharing simple.

4. Scalability

- Scalability shows how well the software handles increased users or workload.
- Performance should not decrease when traffic increases.

Example:

- An e-commerce website handles millions of users during sale periods smoothly.

5. Security

- Security protects software data from unauthorized access.
- Sensitive information is stored in an encrypted and safe manner.

Example:

- A password manager stores passwords in encrypted form for authorized users only.

6. Portability

- Portability means the software can run on different platforms.
- Minimal changes are required when the operating system changes.

Example:

- Games like Minecraft run on Windows, macOS, and Android with similar features.

Funny **R**abbits **U**se **S**mart **S**afe **P**hones

Steps In Software Development Process

1. Requirement Analysis

- In this step, the team discusses with the client and users.
- The goal is to understand what the software should do.

Example:

- For a school management system:
  - Student registration
  - Attendance tracking
  - Result generation

2. Planning

- Planning decides the project scope and workflow.
- Time, cost, resources, and team responsibilities are planned.

Example:

- Deciding:
  - Project duration (3 months)

    - Number of developers and testers
    - Tools and technologies to use

3. Design

- The structure and appearance of the software are designed.
- System architecture and user interface are prepared.

Example:

- Designing:
    - Dashboard layout
    - Menu navigation flow
    - Database structure

4. Development (Coding)

- This is the coding phase where the actual software is built.
- Developers write code based on the design.

Example:

- Writing code for:
    - User profile creation

- Data storage and retrieval
- Report generation

5. Testing

- The software is tested to find and fix errors.
- It ensures the system works as expected.

Example:

- Checking whether:
- Forms accept valid input
- Error messages show correctly
- Features work without crashing

6. Deployment

- After testing, the software is released for users.
- It is installed on servers or app stores.

Example:

- Uploading a mobile app to the Play Store
- Making the system available on a company network

7. Maintenance

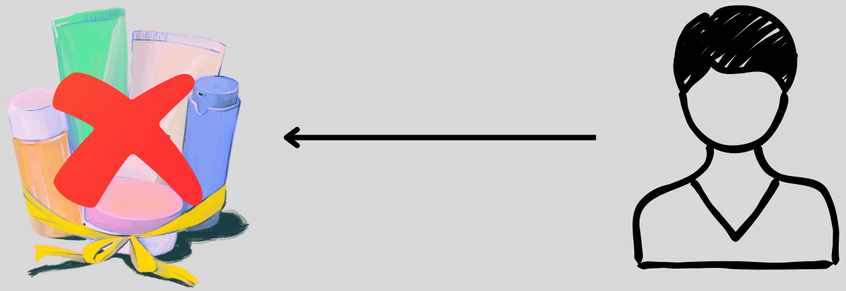
- After release, regular updates and fixes are done.
- New features are added based on user feedback.

Example:

- Fixing reported bugs
- Improving speed
- Adding a dark mode feature

Types of product based on criticality

1. Non - critical products



2. Business-Critical Products

- These products are essential for daily business operations.
- Failure can cause financial loss and disrupt work.

Example:

- Payroll management software – if it fails, salary processing gets delayed.

3. Mission-Critical Products

- These products are required to complete a specific mission or task.
- Failure leads to mission or goal failure.

Example:

- Emergency response dispatch system – if it fails, help may not reach on time.

4. Safety-Critical Products

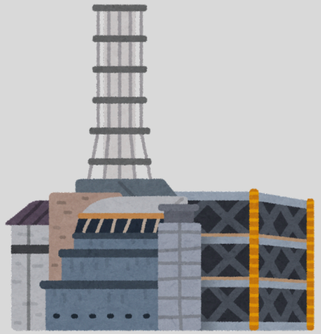
- Failure of these products can create risks to human health or safety.
- High reliability and strict testing are required.

Example:

- Fire alarm system – failure may delay evacuation during an emergency.

5. Life-Critical Products

- These are the most critical products.
- Failure can directly cause loss of life or major accidents.



Example:

- Nuclear reactor monitoring system – failure can lead to catastrophic events.

Problematic Areas Of SDLC

1. Unclear Requirements

- This is the most common problem in SDLC.
- When client requirements are not clear, the development team gets confused.
- Frequent changes in requirements cause rework and delays.

Impact:

- Confusion
- Extra work
- Project delay

2. Ineffective Planning

- Poor planning leads to unrealistic schedules and wrong resource allocation.
- Budget and timeline may exceed the limits.

Impact:

- Late project delivery
- Increased cost
- High team pressure

3. Poor Communication

- Lack of communication between team members or with the client creates problems.
- Misunderstandings result in incorrect implementation.

Impact:
- Wrong features developed
- Rework
- Client dissatisfaction

4. Technical Challenges

- Using outdated technology or selecting the wrong tools causes issues.
- Compatibility and integration problems may occur.

Impact:
- Performance issues
- Integration failures
- Reduced software quality

5. Testing Bottlenecks

- Insufficient time or resources are allocated for testing.
- Due to tight deadlines, testing is rushed.

Impact:

- Bugs remain in the software
- Critical errors in production

6. Deployment Issues

- Problems may occur during deployment due to server or configuration errors.
- These issues affect the software launch.

Impact:

- Failed or delayed launch
- System downtime
- User dissatisfaction

Software Quality Management

Software Quality Management

- Software Quality Management ensures that all activities, inputs, and processes are managed properly.
- The goal is to make sure the final software product meets the defined quality standards.

Levels of Problems in Software Quality Management

1. Correction (Quality Control)

- Correction means fixing defects immediately when they are found.
- It happens during testing or inspection.
- This is a reactive approach.

Example:

- A tester finds a login error and the developer fixes it immediately.

2. Corrective Actions

- Corrective action focuses on finding the root cause of a defect.

- Analysis is done to understand why the problem occurred.
- The goal is to ensure the same problem does not happen again.

Key Points:

- Problem has already occurred
- Action is organized and documented
- Follow-up is done to verify the solution

Example:

- Repeated login failures occur → team analyzes the cause → updates authentication logic to prevent future failures.

3. Preventive Actions

- Preventive actions are taken before a problem occurs.
- This is a proactive approach to quality management.
- It involves monitoring processes, inputs, and performance trends.

Example:

- Reviewing code standards and conducting early design reviews to avoid future defects.

Why Software Has Defects?

What is a Software Defect (Bug)?

- A software defect or bug is an error or flaw in a program.
- It causes the software to give incorrect results or behave unexpectedly.
- A defect prevents the application from working as it should.

Main Reasons for Software Defects

1. Human Errors

- The most common reason for software defects is human mistakes.
- Errors can occur during design, coding, or decision-making.

2. Unclear or Incorrect Requirements

- If the software does not meet user requirements, it is considered defective.
- Misunderstanding or misinterpreting customer needs leads to defects.

3. Poor User Experience

- If the application does not look good or does not feel right to users, it has defects.
- Even if the software works technically, poor usability is also a defect.

4. Defects Can Occur in Any SDLC Stage

- Defects are not only caused by developers.
- Every stage of SDLC involves people, and people can make mistakes.

5. Defects in Requirement Stage

- Errors occur during communication with the customer.
- Requirements may be incomplete, ambiguous, or wrong.

6. Defects in Design Stage

- Poor architectural or system design causes future problems.
- Inexperienced or careless designers may choose incorrect structures.

7. Defects in Development Stage

- This stage produces the highest number of defects.
- Coding mistakes such as logic errors or syntax issues are common.

Quality Management System (QMS)

What is QMS?

- A Quality Management System (QMS) is a set of processes, policies, and practices.
- Its main goal is to ensure that products or services always meet customer expectations and quality standards.

Purpose of QMS

- QMS works as a framework that helps an organization to:
  - Improve operations
  - Increase efficiency
  - Deliver high-quality products or services

How Does QMS Work?

- QMS includes quality planning, quality control, and quality improvement.
- It focuses on quality at every stage of a project or product lifecycle.

Focus of QMS

- Fulfilling customer requirements
- Following legal and regulatory standards
- Improving overall organizational performance

Common Components of QMS

- Quality policies
- Quality objectives
- Standard Operating Procedures (SOPs)
- Continuous improvement practices

QMS in Software Development (Example)

- **In software development, QMS includes:**
  - Code reviews
  - Testing procedures
  - Following development standards (such as ISO 9001 or CMM)

This ensures that the final software is:

- Reliable
- User-friendly

Pillars Of Quality Management System (QMS)

Pillars Of Quality Management System (QMS)

1. Customer Focus

- The main goal of QMS is customer satisfaction.
- Organizations must understand customer needs and try to exceed expectations.

Example:

- Developing software that is easy to use and solves customer problems.

2. Leadership Commitment

- Strong leadership sets the vision and quality goals.
- Leaders create a culture that supports quality across the organization.

Example:

- Management ensures that quality standards are followed by all teams.

3. Engagement of People

- A successful QMS involves everyone in the organization.
- Employees must understand their roles and responsibilities.

Example:

- Developers, testers, and support teams all contribute to product quality.

4. Process Approach

- Work should be done through well-defined processes and workflows.
- This ensures efficient use of resources and consistent results.

Example:

- Following a defined Software Development Life Cycle (SDLC).

5. Continuous Improvement

- Organizations should regularly review and improve their processes.
- Weak areas are identified and improved continuously.

Example:

- Regular software updates, bug fixes, and performance improvements.

6. Evidence-Based Decision Making

- Quality-related decisions should be based on data and facts.
- Decisions should not be made only on assumptions.

Example:

- Making changes based on user feedback and test reports.

7. Relationship Management

- Strong relationships with customers, suppliers, and partners are important.
- This helps in achieving long-term success.

Example:

- Working with reliable vendors and maintaining customer trust.

Important Aspects Of Quality Management System (QMS)

1. Customer Satisfaction

- The main focus of QMS is to meet customer needs.
- Customer feedback and complaints are handled properly.
- The final product should match customer expectations.

2. Quality Planning

- Quality goals and standards are defined in advance.
- It provides a clear plan to achieve quality.

Example:

- Defining testing standards in software development.

3. Process Optimization

- QMS improves processes to make work efficient.
- The aim is to reduce errors and waste.

Example:

- Improving coding and testing workflows.

4. Employee Involvement

- Employees at all levels are involved in quality activities.
- Everyone understands their role and responsibility.

5. Continuous Improvement

- The organization always works to improve quality.
- Focuses on improving processes, products, and services.

6. Risk Management

- Possible risks are identified early.
- Preventive actions are taken to avoid quality problems.

7. Compliance with Standards

- Industry standards and regulations are followed.
- Ensures products are legal and meet quality requirements.

Example:

- Following ISO quality standards.

8. Data-Driven Decisions

- Quality decisions are based on data and analysis.
- Assumptions are avoided.

Example:

- Using defect reports and customer feedback for improvements.

Quality Plan

1. Definition

- A Quality Plan defines the specific quality goals for a product or service.
- It acts like a blueprint that explains how quality will be achieved and maintained.

2. Responsibilities

- The plan clearly defines who is responsible for quality-related activities, such as:
  - Inspections
  - Testing
  - Process reviews

3. Standards and Guidelines

- The quality plan lists quality standards, guidelines, and benchmarks.
- These ensure the final product meets required quality levels.

4. Methods and Workflows

- It describes the methods and workflows to be followed to achieve quality.
- This helps teams work in a consistent and organized way.

5. Risk Management

- The quality plan identifies potential quality risks.
- It also defines their impact and mitigation strategies.

6. Continuous Improvement

- A good quality plan identifies improvement areas.
- It helps refine processes for future projects.

7. Simple Analogy

- A quality plan is like a recipe:
  - What needs to be done
  - Who will do it
  - How it will be done
  - How success will be measured

Quality Control

What is Quality Control?

- Quality Control means inspecting and testing products during production or before delivery.
- Its purpose is to ensure that the product meets required quality standards and specifications.

Role of Quality Control

- Identify defects or quality problems
- Take corrective actions to maintain consistent quality
- Build customer trust
- Reduce waste and improve operational efficiency

Methods of Quality Control

1. In-Process Quality Control

- In this method, products are checked and tested at each stage of production.
- Problems are identified early instead of at the end.

Goal:

- Stop defective products from moving to the next stage.

Example:

- Testing car brakes during assembly to avoid safety issues later.

2. Consignment-Wise Inspection

- In this method, the final batch or consignment is inspected before shipment.
- Ensures that the entire batch meets quality standards.

Inspection Methods:

- 100% inspection of all products
- Sample-based inspection

Example:

- Inspecting a batch of garments for stitching and fabric quality before dispatch.

ISO 9001

What is ISO 9001?

- ISO 9001 is an international standard for Quality Management Systems (QMS).
- It helps organizations deliver consistent quality products and services.
- It focuses on meeting customer expectations and regulatory requirements.

Importance of ISO 9001

1. Consistency in Processes

- ISO 9001 ensures that processes are standardized and repeatable.
- This reduces errors and defects.

Example:

- Following the same testing process for every software project.

2. Focus on Customer Requirements

- The standard emphasizes understanding and fulfilling customer needs.
- Development and testing are aligned with actual user requirements.

Example:

- **Designing test cases based on user requirements.**

3. Continuous Improvement

- ISO 9001 encourages analyzing performance data.
- Processes and quality are improved over time.

Example:

- Improving testing methods using bug reports and test results.

4. Enhanced Documentation

- Proper documentation is mandatory under ISO 9001.
- Quality processes and results are clearly recorded.

Example:

- Maintaining test plans, test cases, and test reports.

5. Global Recognition

- ISO 9001 certification is recognized worldwide.
- It increases trust, credibility, and business opportunities.

Example:

- Clients prefer working with ISO 9001 certified companies.

Practical Example (Software Testing)

- An ISO 9001–certified company follows:
  - Unit testing
  - System testing
  - Proper documentation
  - Defined quality processes

This ensures the final software is reliable and high quality.

Capability Maturity Model (CMM)

What is CMM?

- CMM is a framework that helps organizations improve software development and management processes.
- It shows the maturity level of an organization's processes.

Five Levels of CMM

1. Level 1 – Initial

- Processes are not defined.
- Success depends on individual effort, not on process.
- Projects are inconsistent.

Example:

- A developer works in their own way without any fixed process.

2. Level 2 – Repeatable

- Basic processes are defined.
- Successful practices can be repeated in similar projects.

- Project management basics like scheduling and tracking start.

Example:

- Using a previous project plan to manage a new project.

3. Level 3 – Defined

- Processes are standardized and documented across the organization.
- Teams follow the same procedures.
- Focus on collaboration and quality standards.

Example:

- Company-wide coding standards and testing process are followed.

4. Level 4 – Managed

- Processes are measured and controlled using data and metrics.
- Performance is quantitatively tracked.

Example:

- Tracking defect rates, test coverage, and productivity metrics.

5. Level 5 – Optimizing

- Highest maturity level.
- Focus on continuous process improvement and innovation.
- Problems are solved proactively.

Example:

- Adopting new tools, automation, and continuously improving processes.

Limitations of cmm

1. Focus on Processes, Not Outcomes

- CMM emphasizes process maturity rather than actual project results.
- Following processes does not guarantee a high-quality product.

Example:

- A team may follow all steps, but the final software may still have issues.

2. Costly and Time-Consuming

- Achieving higher CMM levels requires time, resources, and investment.
- Small organizations may find it difficult to implement.

3. Rigid Structure

- CMM has a linear, one-size-fits-all structure.
- It may not fit the unique working style of every organization.

4. Limited Focus on People

- Emphasizes processes and tools over team skills, motivation, and creativity.

5. Overemphasis on Documentation

- Extensive documentation is mandatory.
- Sometimes it can be overwhelming and counterproductive.

6. Not Industry-Specific

- CMM is a generic model.
- It does not provide industry-specific guidance or practices.

Thank You