Software Quality Assurance and Testing

Table of Contents

Chapter 1: Process Assurance

Definition of Testing

1. Testing is the process of executing a program or product with the intent of finding errors.
2. Testing is the process of demonstrating that a program or product does what it is supposed to do.

Why is Testing necessary?

1. Bug detection
2. Quality improvement
3. User satisfaction
4. Prove that the software or application has no errors

Software Development Costs

1. Development (33%)
   - Requirement
   - Design
   - Code
2. Diagnosis & Repair (57%):
   - Debugging
   - **Regression Testing**: It makes sure that previously working functionality still works, after changes elsewhere in the system.
3. Testing (10%)

Process Assurance

1. Definition of **Process Assurance**: The activities to ensure process used is integrated, consistent, and correctly applied.
2. **System Development Life Cycle (SDLC)**
   - Requirement
   - Design
   - Code

- Testing
- Maintenance

3. **Steering Committee**

   A steering committee is responsible for

   (1) Defining project policy.
   (2) Reviewing milestones.
   (3) Estimating the time that will be required to maintain the system.
   (4) Evaluating the risk factors.
   (5) Deciding on the type of support required.
   (6) Deciding when the data will be available and used.
   (7) Forming a Configuration Control Board (CCB) that manages the impact of changes.

4. **Project risks**
   - Business risks
   - Technology risks
   - Project size risks

Causes of Failure in Process Assurance

1. Lack of management support.
2. Lack of user Involvement.
3. Lack of project leadership.
4. Lack of measures of success.

Verification vs. Validation

1. **Verification**
   - **Process**-related activities.
   - Throughout the SDLC, verification activities take place continuously to ensure that at any phase, a product is being built according to the standards and requirements.
   - Example: walkthrough, reviews, inspections, audits and official “sign-offs”
2. **Validation**
   - **Product**-related activities such as testing.
   - The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements.
   - Example: unit testing, system testing, acceptance testing

Product Delivery Process

Market-Oriented Life-cycle:

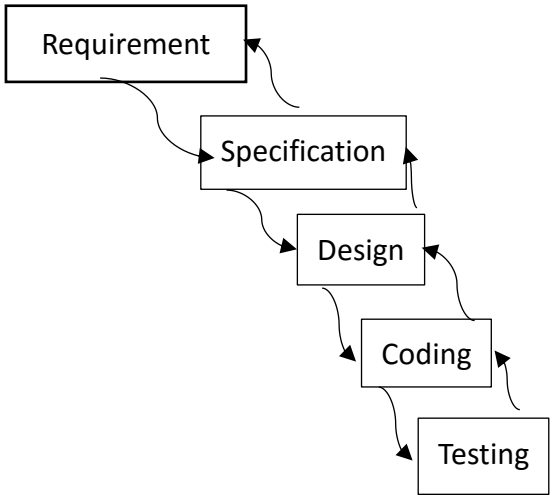
- Requests for proposals
- Business plans
- Corporate mission statements
- Contracts to build custom software

Parallel Product Development

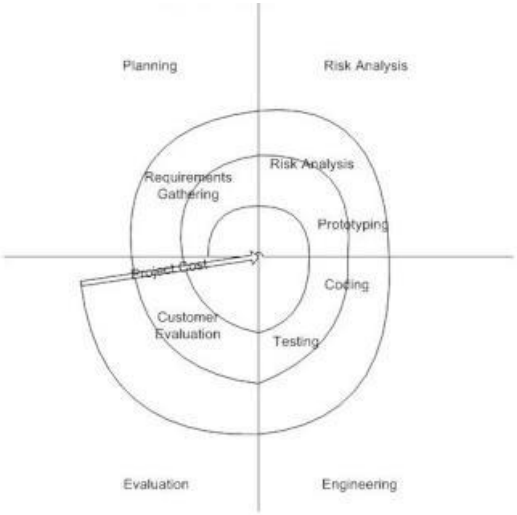
When the software components are being built, the documentation, training, sales, delivery and support components are being built

Waterfall

Waterfall is a sequential software development process



Spiral Lifecycle



The **spiral model** is a **risk-driven** process model generator for software projects, which is similar to the incremental model. It has four phases:

- Planning
- Risk Analysis
- Engineering
- Evaluation

Agile

1. **Agile** is an iterative and incremental approach to software development, which is performed in a collaborative manner.
2. It helps teams respond to the unpredictability and minimizes overall risks, but it lacks the overall design.
3. **Sprints**: A sprint is a get-together of people involved in a project to further a focused development of the project.

Ad Hoc Team

1. An **ad hoc team** is one in which teammates must work together to obtain a common goal, but without any prior agreement regarding how to work together.
2. **Filters**
   - They prevent defects from going to subsequent phases.
   - Examples: checklists, templates, outlines or prototype documents.

Chapter 2: Product Assurance

White Box Testing

1. **White box testing** tests the internal structure of the program. It's also called program testing and glass box testing.
2. **Unit Testing**: A unit test is a piece of a code (usually a method) that invokes another piece of code and checks the correctness of some assumptions afterward.
3. Definition of **Unit**
   - Smallest piece of software testable in isolation.
   - In procedural programming, may refer to a single function or method.
   - In object-oriented programming, may refer to a single class.
4. **Basic path testing** is a structured testing or white box testing technique used for designing test cases intended to examine all possible paths of execution at least once.

Black Box Testing

1. **Black box testing** tests the functionalities of the program without looking into its internal structure.
   - Assert expected output based on given inputs
2. **System Testing** validates a program by checking it against the published user or system requirements.
   (1) **Volume testing** subjects the program to heavy volumes of data.
   - Volume Testing = Large amounts of data
   - Load Testing = Large amount of users
   - Stress Testing = Too many users, too many data, too little time and too little room
     Example: You conduct stress test for 5000 users accessing the system by testing with the following number of users: 1, 50, 500, and 2500, 5000.

   (2) **Functional testing** verifies a program by checking it against the requirement documents or specifications.
   (3) **Security testing** makes sure that the system only can be accessed by authorized users.
   (4) **Recovery testing** determines how quickly the system can recover after it has

gone through a system crash or hardware failure.
Example: You can conduct recovery testing by simulating a power failure.

(5) Documentation testing tests the documented artifacts that are usually developed before or during the testing of Software.

(6) Environment testing, also known as Integration testing, verifies a piece of equipment can withstand the difficult conditions of harsh environments.

JAD (Joint Application Development)

1. **JAD** is a process where the customer is involved in the application development by a series of workshops.
   - Member: key users and technology analysts
   - Purpose: accelerate requirement gathering and design process
   - User: a set of similar users, usually in the same company
   - ID: requirements, procedures, problems, system proposals, and outputs
2. The benefits of conducting a **JAD session**:
   - It can result in obtaining proper requirements.
   - Users have a better understanding of the system.
   - Any constraints in the development of the system are brought to the forefront.
   - The expectations of the users are managed and user ideas are discussed upfront.

JAD Agenda

The JAD session Agenda consists of

- summary of the project by the sponsor
- gathering and evaluation of the user requirements
- implementation strategy
- signoff

Prototyping

1. A **prototype** is a mock-up, interactive model that is developed in less time with less expense in the requirements phase.
2. It determines how various functionalities and interfaces of the system would work and shows the users an early version of the system to get their feedback.

Disaster Recovery

1. **Hot site**: A hot site is an off-site computer center equipped with hardware where a company can move its operations.
2. **Cold site**: A cold site is an empty room or building which doesn't have any hardware.
3. Electronic vaulting is where copies are placed in the vault electronically after a scheduled backup and is available for immediate retrieval.
4. Network recovery
5. Testing
6. Training
7. Approval

Configuration Management

1. **Configuration management** determines clearly about the items that make up the software or system. These items include source code, test scripts, third-party software, hardware, data and both development and test documentation.
2. It's used for
   - keeping an inventory of company's products, documentation, source code, and test plans.
   - controlling projects, documents, and overall CM process.
   - code management.

Why is Management Support Important?

1. Strategic Planning
2. Motivating staff
3. Evaluating to see if a change of direction is required
4. Providing atmosphere to succeed.

Benchmarking

1. **Benchmarking** is the process of comparing a company's business processes and performance metrics to the ones of other companies.
2. Determine the problem -> Decide on what data to benchmark -> Collect data -> Analyze data -> Compare performance -> Improvement -> Adapt and implement -

> Continuously monitor results

Chapter 3: Software Quality Assurance

The absence of SQA:

- Testing is each individual's job.
- The testing specialist assigned to each project.
- Review committee.
- Testing is a part of the development.

Cost of Poor Quality (CoPQ)

1. Analysis of defects
2. Rework
3. Losing opportunities
4. A Loss of revenue
5. Poor customer satisfaction

Chapter 4: Software Quality Standards

Six sigma

Six sigma is a strategy to reduce errors to less than 3.4 defects per million lines of code, which includes the following six steps:

1. Determining **what** your product is.
2. Determining **who** the customer is for the product.
3. Identifying the **suppliers** you need for your product.
4. Mapping out the **process** you must use to put it together.
5. **Examining the process** to remove errors and wasted steps.
6. **Establishing measurement** means to feed continuous improvement.

Product Delivery Process

Deliverables associated with a PDP:

- Marketing analysis & positioning plan
- General product description
- Marketing demonstrations

- Preliminary & final sales plans
- Preliminary budget
- Requirements definition
- System specification
- Software architecture
- Project plan
- Product development
- Software audit
- Behavioral testing
- User profile testing (workflow/tasks/products)
- Marketing documents
- Training, support & delivery plans

Release Management

Release management is the process of managing, planning, scheduling and controlling a software build through different stages and environments; including testing and deploying software releases.

- Product development and filters
- Entry and Exit Criteria
    - **Entry Criteria** determine when a given test activity should start.
    - **Exit Criteria** determine whether a given test activity has been completed or not.
- Product Announcement

Chapter 5: Overview of Test Cycles

Objectives of Testing

1. Quality awareness.
2. Understanding of customers' pains and experience.
3. Closed loop learning through RCA (Root Cause Analysis).
4. Understanding of which processes are working and which needs to be changed.
5. Awareness of good developers.
6. Identifying defects before your customers see them.

Black Box Testing

1. **Error Guessing**
2. **Equivalence Partitioning** (EP): considering test cases from each partition.
   Example: Input = [-10, 20] -> Test cases = {-15, 5, 25}
3. **Boundary Value Analysis** (BVA): considering test cases from each partition and boundary values.
   Example: Input = [-10, 20] -> Test cases = {-15, -10, 5, 20, 25}
4. **Cause-and-Effect Graphing**: dependencies.

Integration Testing

1. **Integration testing** tests integration or interfaces between components, interactions to different parts of the system.
2. There are nine software integration perspectives:
   (1) Incremental
   (2) **Big Bang**: tests all components or modules which are integrated simultaneously, after which everything is tested as a whole.
   (3) **Top-down**: tests the highest-level modules and then tests the branch of the module step by step until the end of the related module.
   (4) **Bottom-up**: tests the lowest-level modules first and then uses them to make the testing of higher-level ones easier.
   (5) **Risk-based**: is an approach where the integration testing is performed starting with the risky and hardest software module first.
   (6) Threaded
   (7) Outside-in
   (8) Inside-out
   (9) Little bang
3. **Beta Testing**: A testing carried out by real users in the real environment.
4. **Regression Testing**: It makes sure that new changes won't break existing functionalities.
5. **Acceptance Testing**: confirms that the system is developed according to the user requirements and is ready for operational use.

Chapter 6: Test Planning

1. Benefits of A Test Plan

- Establishes a test schedule.
- Acts as a service agreement between testers and developers.
- Identifies what will be tested and risks involved in testing.
- Communicates
    - The method which will be used to test the product.
    - Required resources: hardware, software, and manpower.
    - Testing schedule.
    - The process for managing the testing project.

2. Contents of A Test Plan
    - Objective
    - Functions to be Tested
    - Functions Not to be Tested
    - Approach
    - Environment Requirement
    - Test Schedule
    - Resources and Responsibilities
    - Dependencies
    - Risk Analysis
    - Automated Tools Used
    - Reviews
    - Entry and Exit Criteria
    - Approvals

Evaluation of Automated Test Tools

1. **Compatibility** – Does the tool support the platform(s) and environment(s) of the system under test?
2. **Usability** – Does it provide the necessary functionality, and are the testers able to make effective use of it without a prohibitive learning curve?
    - Functionality
    - Extensibility
    - Learning Curve
3. **Maintainability** (of tests/test cases/test data) - Can changes to the system under test be accommodated in the automated tests without too much effort?
4. **Manageability** (of library and measurements) - Does the tool provide sufficient and meaningful information to enable management of the test library and measurement of the test results?

Type of Automated Test Tools

1. **Program-Logic Analyzer:** reads the programs as inputs and produces reports to help you understand the logic of the code and how the program performs.
2. **Test coverage tool**: produces reports to help you analyze all the branches of the code that may have/have not been exercised by your test cases.
3. **Test driver**: provides features that offer a specific tool "language" to develop the test cases and test data to be exercised.

Outsourcing

Outsourcing is when a company decides to have another company build their software systems for them.

Chapter 7: Software Quality Assurance Reviews

What is a review?

1. A review is a process or meeting during which a software product is examined by a project personnel, managers, users, customers, or other interested parties for comment or approval.
2. A review is an evaluation of a deliverable.

Formal Review

1. **Deliverable** is a report with roles and responsibilities of the action items.
2. A formal review has the structured review plan and agenda.
3. A typical formal review process consists of six main steps:
   - Planning
   - Kick-off: gets everybody on the same wavelength regarding the document under review and commits to the time that will be spent on checking.
   - Preparation
   - Review meeting
   - Rework
   - Follow-up

Reviews

1. (I) **Walkthrough** is an informal review process where the designer leads one or more others of the project team through a segment design or code and the participants ask questions and make comments about possible errors, violation of development standards, and other problems.
2. (F) The **Inception**s are peer reviews of software which were developed by Michael Fagan in 1976, involving the following steps:
   - Planning
   - Overview
   - Preparation
   - Inspection
   - Rework
   - Follow-up
3. **Technical Review**
4. **Management Review** is conducted by management representatives to evaluate the technical quality of software products.
5. **Peer Review** is a review where a work product (document, code, or other) is examined by its author and one or more colleagues, in order to evaluate its technical content and quality.

Chapter 8: Basic Concepts of Measurements

Definition of Metrics

1. **Metrics** are a set of measurements to quantify results.
2. Performance metrics quantify the unit performance.
3. Project metrics tell you whether the project is meeting its goals
4. Business metrics define the progress of business in measurable terms.

Understanding the Need for Collecting Metrics & Benefits

1. Make fact-based decisions about the quality of the product.
2. Improve productivity and quality of processes.
3. Develop, identify and analyze trends.
4. Decrease development costs.

5. Increase customer satisfaction.

Cost of Metrics

1. Collection: data capture, data storage, tools, and resources.
2. Analysis
3. Trending
4. Training

Popular Metrics

1. **Number of Defects**
   - Total # of Defects Found
   - Type of Severity: Show Stopper, Serious, and Moderate
2. **Work Effort**
   - Number of hours spent on the development
   - System enhancement
   - System support or maintenance
3. **Schedule**: tracks the performance of the project team toward meeting the committed schedule.
4. **Number of Changes to the requirements**: defines the type of change
   - Functionality
   - Design
   - Response time
   - Environment
5. **Size**
   - Line of code
   - Function Points:

     A **function point** is a "unit of measurement" to express the amount of business functionality.
     - **Logical Internal Files**: data stored and maintained by the system
     - **External Interface Files**: data passed or shared with other systems
     - **External Inputs**: transactions, additions, and changes
     - **External Outputs**: reports
     - **External Inquiries**: no updates
6. **Documentation Defects**
   - Type of Documentation

- Nature of error:
    - missing functionality
    - unclear explanation
    - spelling
    - not user-friendly

7. **Code complexity**

   KPI: Key Performance Indicators

Measurement

<u>Process for measurement</u>

Determine -> What is it that you are going to measure?

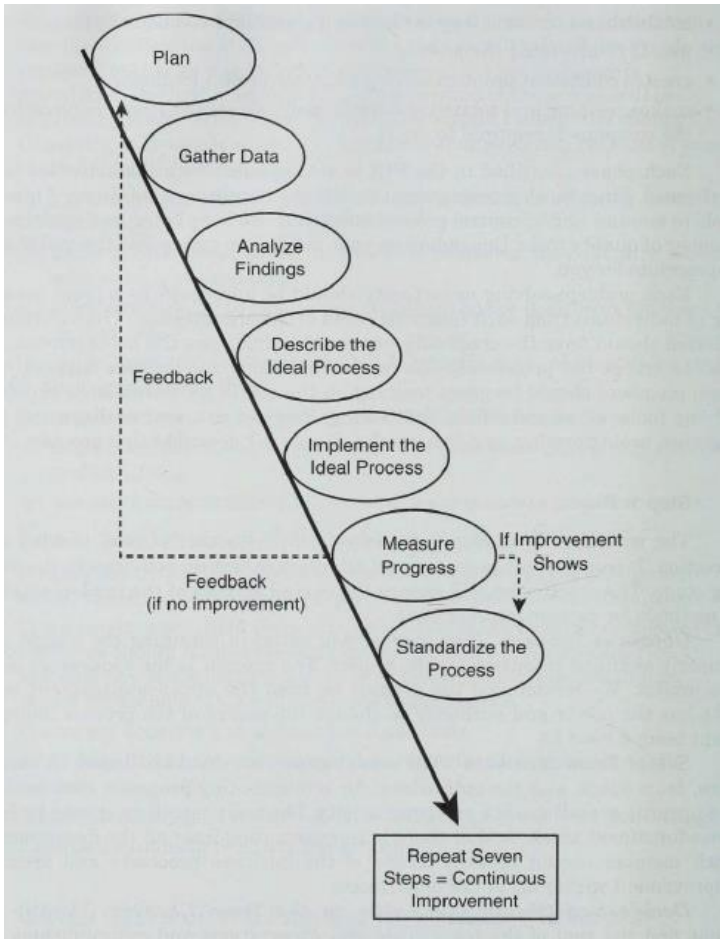
Gather -> Date resource management

Measure -> Use realistic measurement criteria

Correct -> Take corrective actions to improve quality or productivity

Chapter 9: Process Improvement Road Map

Seven Steps to the Process



Quality Tools

1. **Cause & Effect diagram** (= Ishikawa = Fishbone): shows the causes of a specific event.
2. **Scatter diagram** is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.
3. **Process Flow diagram** is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows.
4. **Pareto Chart** is a type of chart that contains both bars and a line graph, where individual values are represented in descending order by bars, and the

cumulative total is represented by the line.

5. **Brainstorming** is a group creativity technique by which efforts are made to find a conclusion for a specific problem by gathering a list of ideas spontaneously contributed by its members.
6. **Checklists** ensure all necessary issues have been identified and addressed.
7. **Surveys** are the methods for collecting information using questions.
8. **One-on-one meetings** are the meetings where only the boss and one other employee are present.

Chapter 10: Standards and Evaluation of Process

What is ISO (International Standards Organization)?

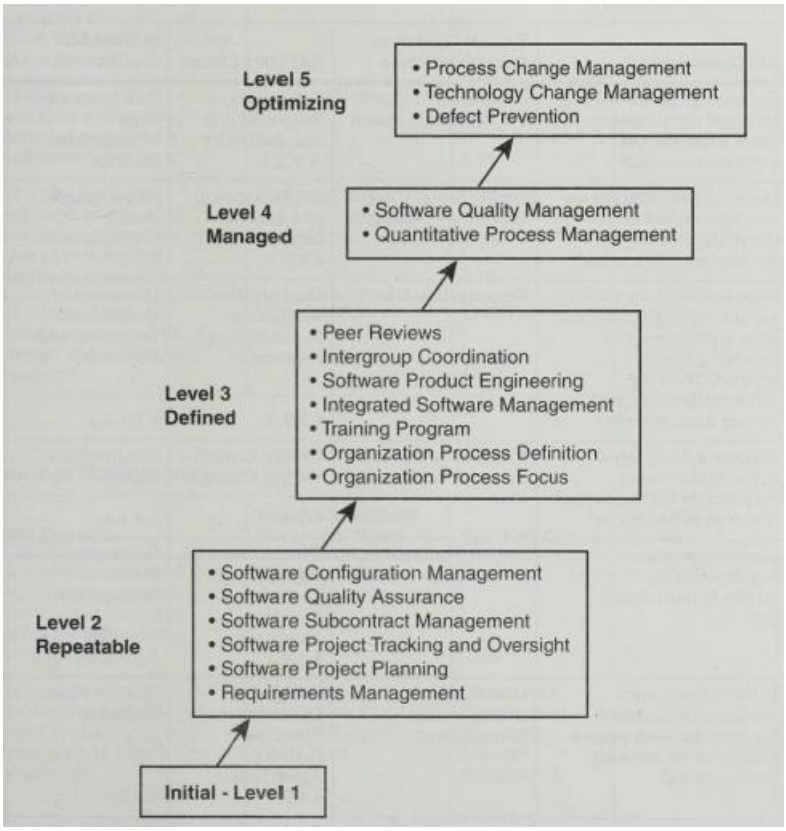
1. **ISO** is the specialized international organization for standardization and has over 100 member countries.
2. American National Standards Institute is the United States member.

ISO 9000 Family of Standards

1. **ISO 9000**: "Quality management and quality assurance standards guidelines." It describes **basic principles and vocabulary** of a quality management system and defines the terminology
2. **ISO 9001**: "Quality Management Systems – **Requirements**." It describes the requirements relative to a quality management system either for internal use or for contractual or certification purposes.
3. **ISO 9004**: "Quality Management Systems – **Guidelines**." ISO 9004, which is intended for internal use and not for contractual purposes, focuses particularly on continually improving performance.

SEI Capability Maturity Model Integration (CMMI)

CMMI is a process improvement approach that helps organizations improve their performance.



Chapter 11: Software Development, Total Quality Management, and Risk Management

Risks

- Missing deadlines
- Poorly defined users
- Creeping featurism
- Low reusability
- Non testability
- Personnel shortages
- Poor Training
- New technology

Risk Management

Minimize the risk through/by

- Error tracking
- RCA
- Automated tools
- Increasing testability
- Creating testable software architectures

Acceptance Criteria are the conditions that a software product must satisfy to be accepted by a user, customer, or in the case of system level functionality, the consuming system.