Software Quality Assurance

Hansika Ukgoda

Evaluation Criteria -

Continuous Assessment : 40%

Final Assessment : 60%

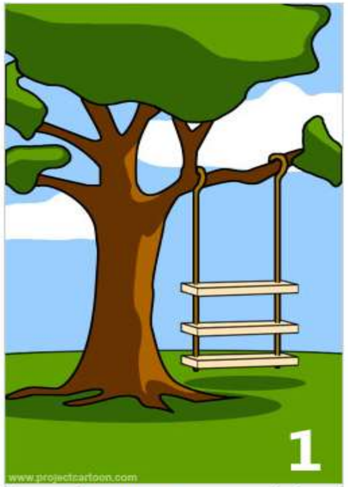
OutLine

- Introduction to the Software Quality Assurance
- Testing Overview
- Testing Methods and types
- Test Management
- Development and Quality Plans
- Standard Procedures and Introduction to Automation Tools

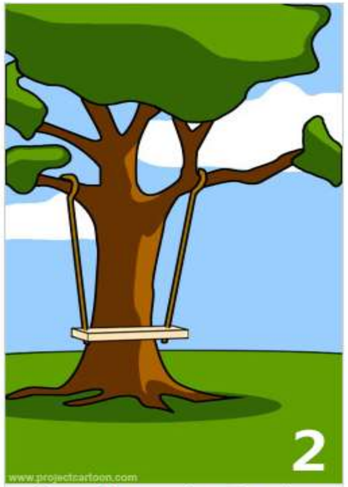
Chapter 1 – Introduction to Software Quality Assurance

Why software has defects ?

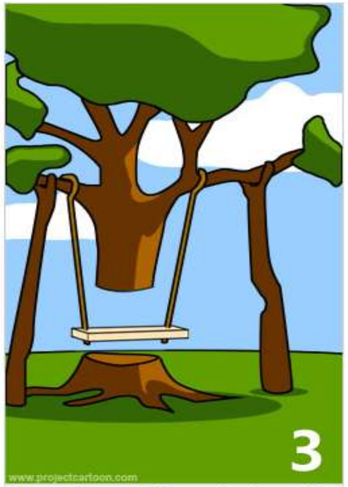
- Customer cannot explain what they really need
- Requirement specification is not clear
- We misunderstood the requirements
- When developing we might do lot of mistakes
- We are under time pressure



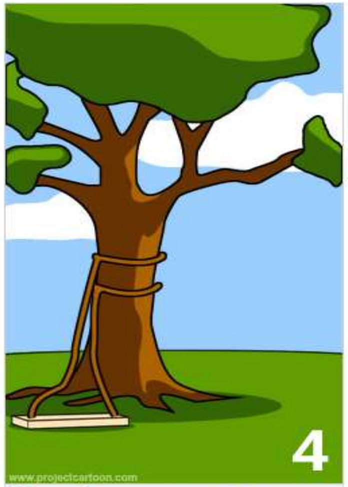
How the customer explained it



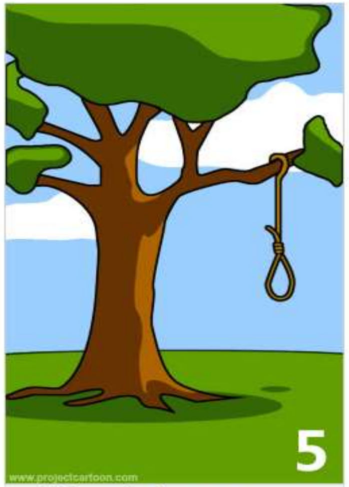
How the project leader understood it



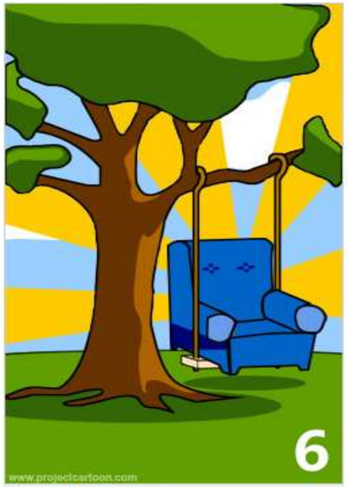
How the analyst designed it



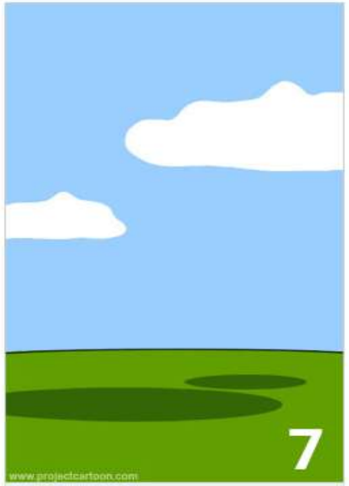
How the programmer wrote it



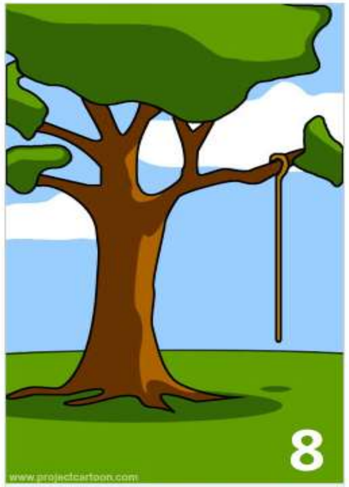
What the beta testers received



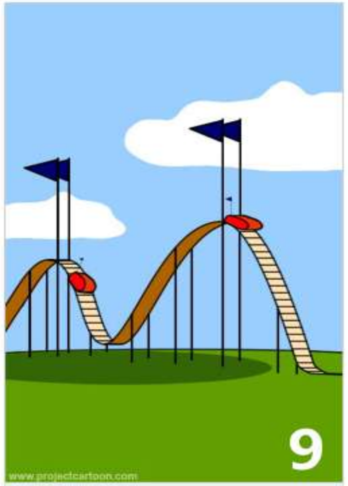
How the business consultant described it



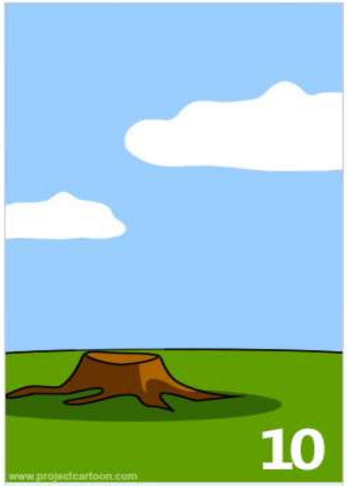
How the project was documented



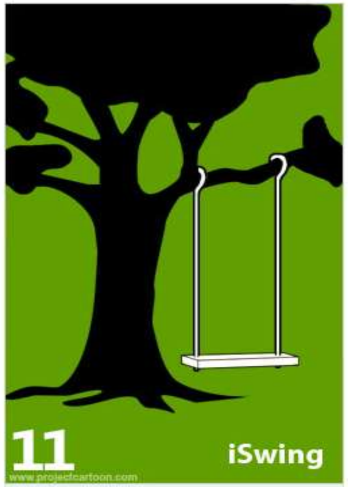
What operations installed



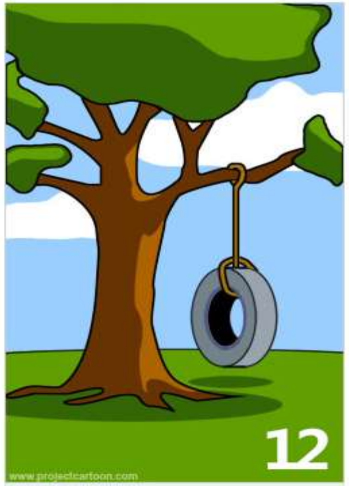
How the customer was billed



How it was supported



What marketing advertised



What the customer really needed

What is Quality?

A product which meet customer requirement and fit to use.

- Transcendent: Quality is something that is intuitively understood but nearly impossible to communicate. Like beauty or love.
- Product Based: Quality is the features and attributes of a product
- User Based: If the customer is happy then it is of good quality.
- Manufacturing Based: If the product conforms to the design specifications, then it have a good quality.
- Value Based: If the product is perceived as providing good value for the price, it is quality product

Definition of SQA:

SQA is a systematic process to ensure that software meets specified requirements, is free of defects, and is developed within the stipulated time and budget.

Key Objectives:

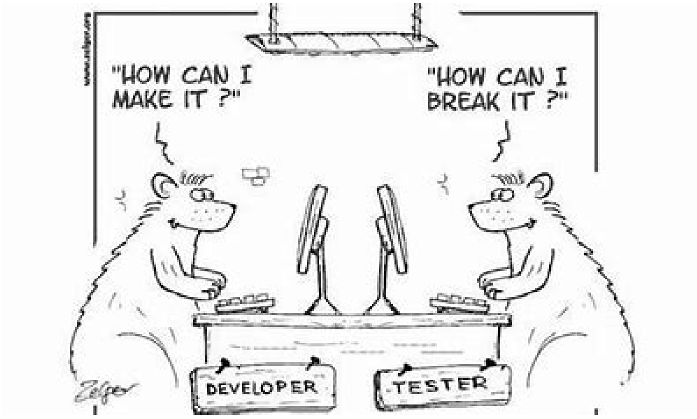
- Delivering reliable and efficient software
- Meeting customer expectations
- Reducing risks and uncertainties

SQA Frameworks?

Testing vs Quality Assurance

- Testing - Detect the defects
- QA - Prevent the defects

QA Psychology



Quality Factors in Software Development

Overview:

Quality factors are characteristics that determine the overall quality of software.

Key Quality Factors:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

Trade-offs in Software Development

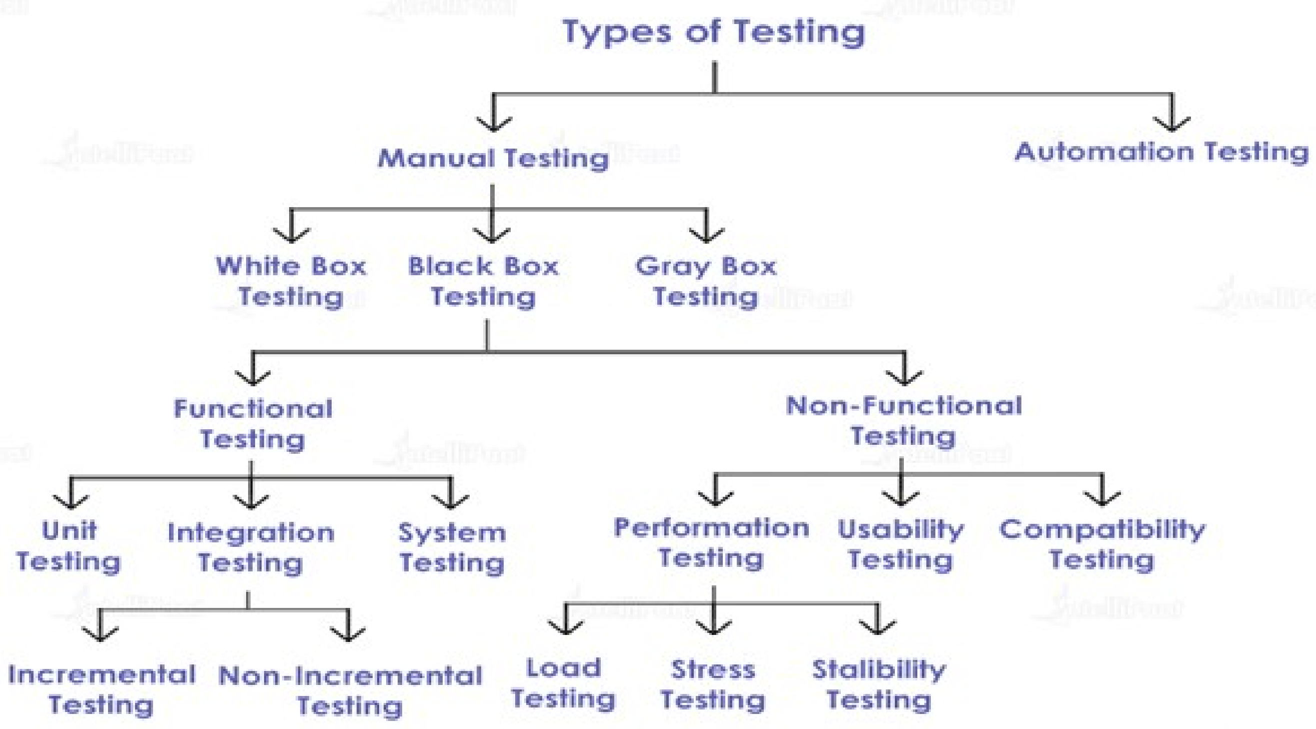
- Functionality vs. Efficiency
- Usability vs. Security
- Maintainability vs. Development Speed
- Portability vs. Resource Utilization
- Reliability vs. Development Costs
- Scalability vs. Simplicity

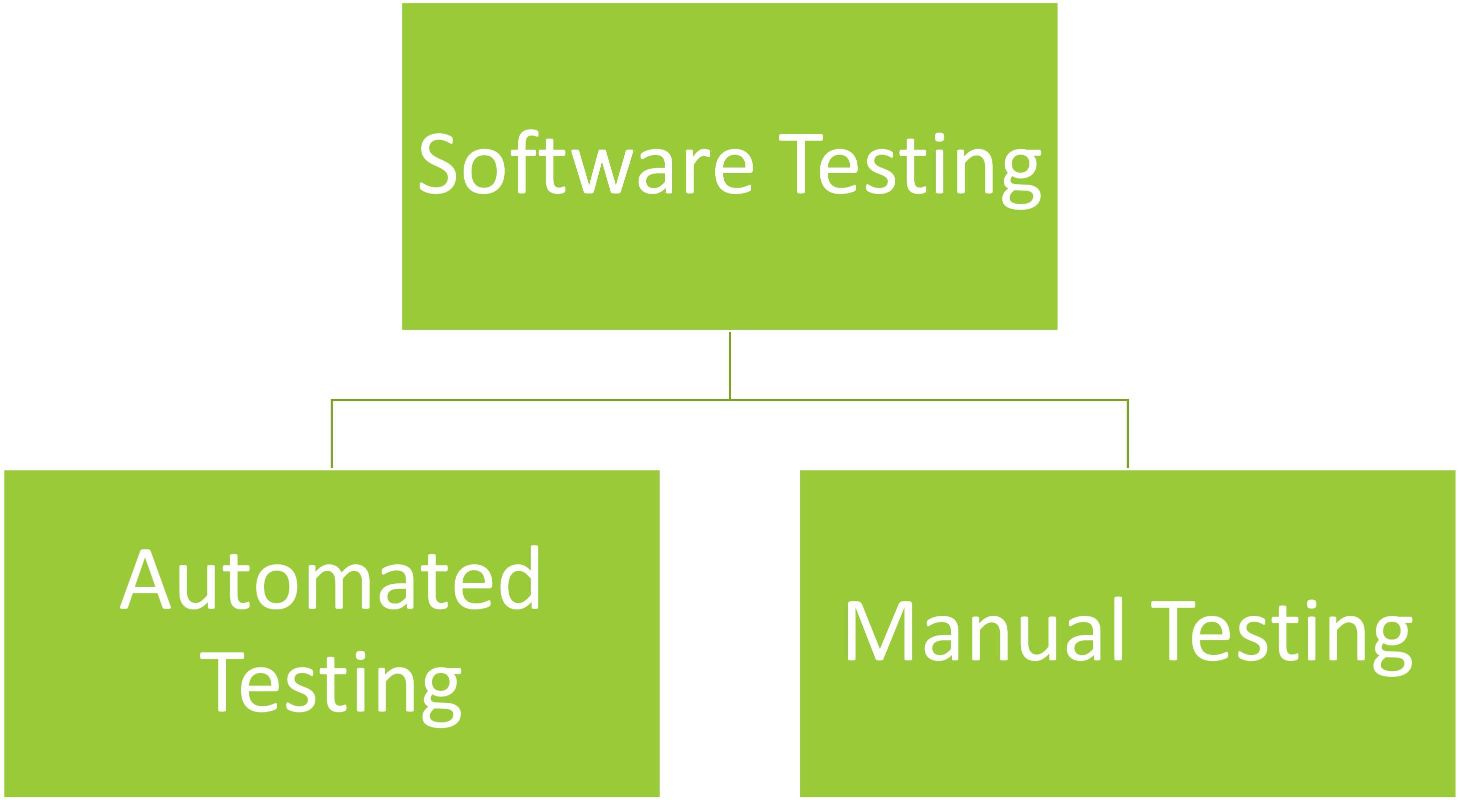
Importance of Software Testing

- Ensures that Software Meets Specified Requirements.
- Identifies Defects and Bugs Early in the Development Process
- Validates the Software's Functionality, Performance, and Security
- Ensures Compatibility Across Platforms
- Verifies Data Integrity and Reliability
- Facilitates Regression Testing for Ongoing Development

Types of Testing:

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing





Software Testing

Automated Testing

Manual Testing

Automated Testing	Manual Testing
Benefits: Efficiency: Faster execution. Regression Testing: Ensures system stability. Consistency: Reduces human error. Limitations: Setup Time: Requires initial time and resources. Non-Visual Verification: Limited UI evaluation.	Benefits: Exploratory Testing: Uncovers unforeseen issues. User Experience Evaluation: Effective for subjective aspects. Cost-Effective for Small Projects: Lower overhead for small projects. Limitations: Subjectivity: Results influenced by tester bias. Resource-Intensive: Time-consuming for large-scale testing. Prone to Human Error: Manual execution may lead to inconsistencies.

Understanding Software Bugs

Definition: A software bug is a coding error that causes a program to behave unexpectedly.

Common Types of Bugs:

- Syntax Errors
- Logic Errors
- Runtime Errors

Root Cause Analysis:

- Purpose:

  Deeper understanding of why bugs occur.

- Process:

  Systematic investigation to identify fundamental causes.

- Benefits:

  Prevents recurrence

  Improves development processes.

Bug Tracking Systems:

Examples:

1. Jira
2. Bugzilla

- Purpose:

  Efficient bug management.

- Features:

  Issue tracking

  Collaboration

  Progress monitoring.

- Benefits: Streamlines communication, ensures issue resolution.

Impact Analysis

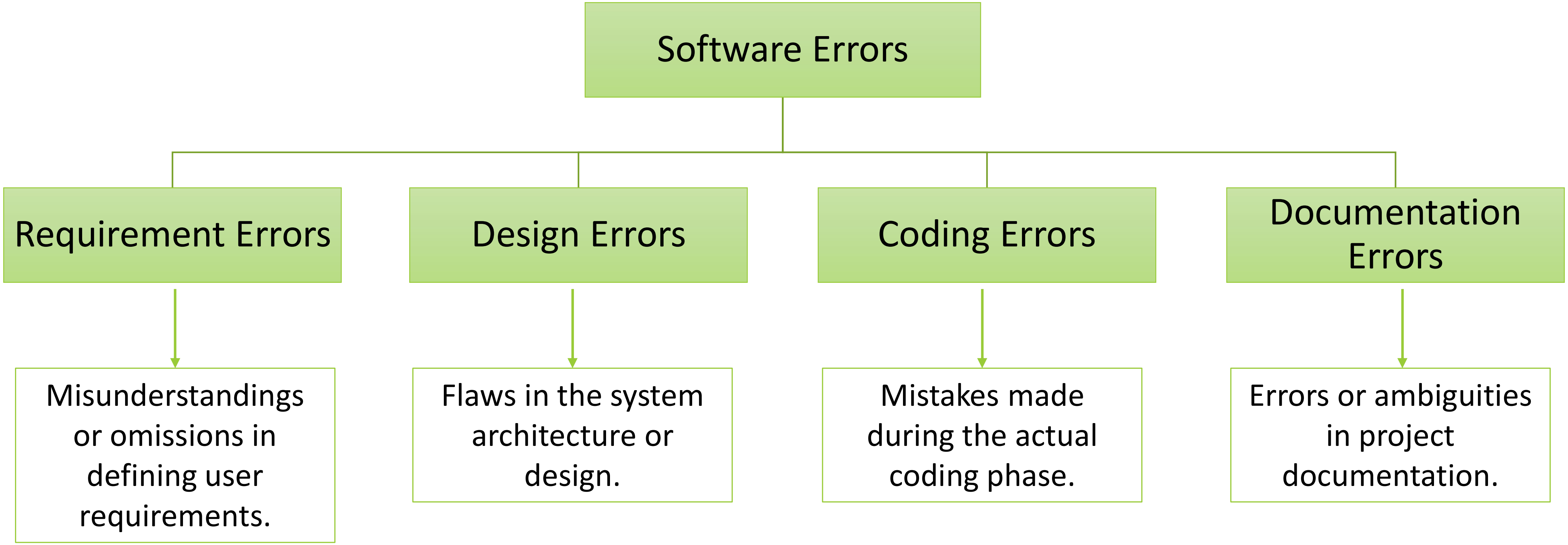
Impact analysis is a systematic process used in software development to assess the potential consequences or effects of a change, defect, or issue on the overall system.

Benefits:

Risk Mitigation: Enables proactive measures to minimize potential negative impacts.

Resource Planning: Helps allocate resources for bug resolution effectively.

Classification of Software Errors



Software Errors

Requirement Errors

Misunderstandings or omissions in defining user requirements.

Design Errors

Flaws in the system architecture or design.

Coding Errors

Mistakes made during the actual coding phase.

Documentation Errors

Errors or ambiguities in project documentation.

industry-standard error classification systems

IEEE Standard: Recognized in the software industry.

Provides a systematic approach to categorize software anomalies.

Purpose -

Consistency: Promotes uniformity in identifying and documenting anomalies.

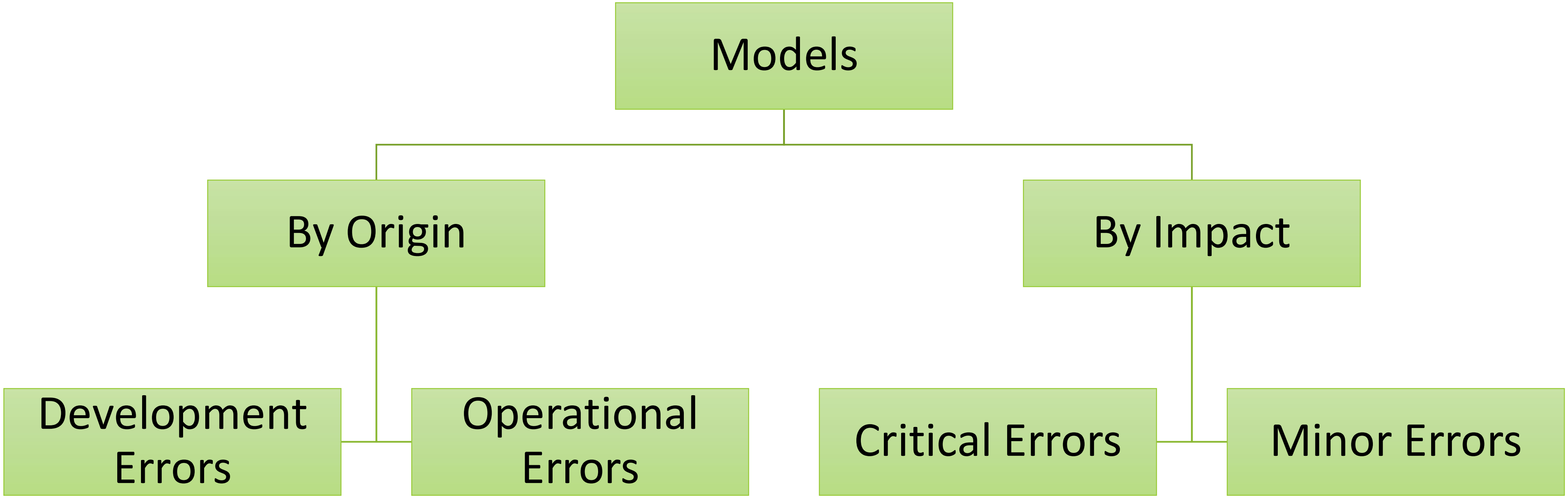
Analysis: Facilitates in-depth analysis of software quality and performance.

Critical Errors and Consequences:

Examples -

- NASA's Mars Climate Orbiter (1999)
- Therac-25 Radiation Therapy Machine (1985-1987)
- Heartbleed Vulnerability (2014)

Common Software Error Classification Models



Models

By Origin

By Impact

Development Errors

Operational Errors

Critical Errors

Minor Errors

Minor Errors and Resolutions:

Examples -

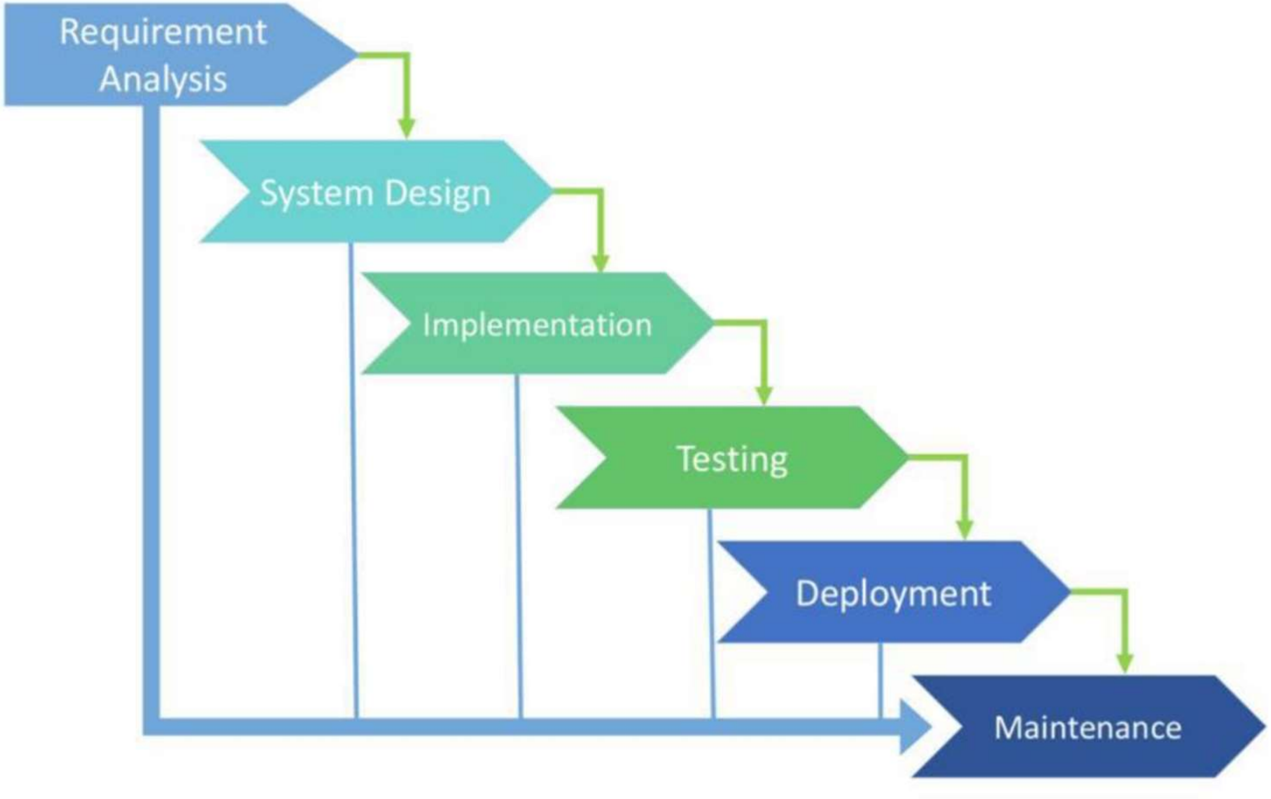
- Microsoft Excel's "Patriot Missile" Bug (1991)
- iPhone "Antennagate" (2010)
- Google's Calendar Sync Bug (2006)

Discuss strategies for preventing each type of error during the software development life cycle.

Strategy ??

Bugs and Consequences ??

SQA Components in the Project Life Cycle



Planning Phase →

Objective: Setting the Foundation for Quality

- Requirements Analysis and Planning: Analyzing project requirements and developing a comprehensive SQA plan.
- Process Definition and Implementation: Defining development processes, coding standards, and testing procedures.
- Training and Education: Providing training to the team to ensure they understand and adhere to defined processes.

Development Phase →

Objective: Integrating SQA into the Development Process

- Configuration Management: Managing and controlling changes to software configuration throughout development.
- Quality Metrics and Measurements: Defining and utilizing metrics to assess the quality of development processes and deliverables.
- Continuous Improvement: Establishing a culture of continuous improvement, learning, and adapting processes.

Testing Phase →

Objective: Ensuring Software Quality through Rigorous Testing

- Testing and Test Planning: Developing a comprehensive testing plan and executing tests at various levels.
- Defect Tracking and Resolution: Identifying, documenting, and resolving defects found during testing.
- Automation Tools and Techniques: Utilizing automated testing tools to streamline and enhance the efficiency of testing.

Deployment and Maintenance Phase →

Objective: Ensuring Post-Deployment Quality and Stability

- Documentation and Reporting: Maintaining comprehensive documentation and providing regular reports on project and quality status.
- Customer Feedback and Satisfaction: Gathering feedback from end-users, incorporating insights, and ensuring continuous user satisfaction.
- Compliance with Standards: Ensuring adherence to industry standards and regulatory requirements post-deployment.

Strategies for Error Prevention

- Clear Communication
- Thorough Testing
- Documentation
- Training and Skill Development
- Collaboration and Code Reviews
- Version Control
- Continuous Integration (CI)

Conclusion

Recap of Key Points:

- SQA ensures high-quality software development.
- Quality factors play a crucial role in determining software excellence.
- Software testing is essential for bug identification and validation.
- Understanding and classifying errors help in effective error management.

QUESTIONS

Thank
You