Software Testing

Overview

Lecturer: Nguyen Thanh Quan (MSc)
Email: tg_nguyenthanhquan_cntt@tdtu.edu.vn

Content

1. Software definition
2. Software quality
3. Software components
4. Software testing
    - Error/bug/fault/failure
    - Software development life cycle
    - Software testing life cycle
    - Software testing terms and methodologies

Software definition

Software comprises the entire set of programs, procedures, resources, data and routines associated with the operation of a computer system that tell a computer how to perform specific tasks.

Software often includes associated software documentation.

Software components (IEEE)

- Computer programs
- Procedures
- Documents
- Data

The importance of computer program

Request and command computer to perform required tasks

The importance of procedure

Determine the sequence, time, methods, and actors performing applications

The importance of document

1. **Documents for software development:**
    - Requirements
    - Design (blueprint)
    - Specifications:

➔ Fosters stakeholders collaboration, verification and validation.

2. **Documents for end users:** provides users with the needful information of using the product.
3. **Documents for software development crew:** provides engineers involved in the development with detailed implementation information.

The importance of data

Parameters, arguments, names etc are customized and adapted to specific users

Testing data

Software development life cycle (SDLC)

- Standard model used worldwide to develop softwares.
- A framework that describes the activities **performed** at **each stage** of a **software development** project.
- Necessary to **ensure** the **quality** of the software.
- **Logical steps** taken to develop a software product.

Software testing - introduction

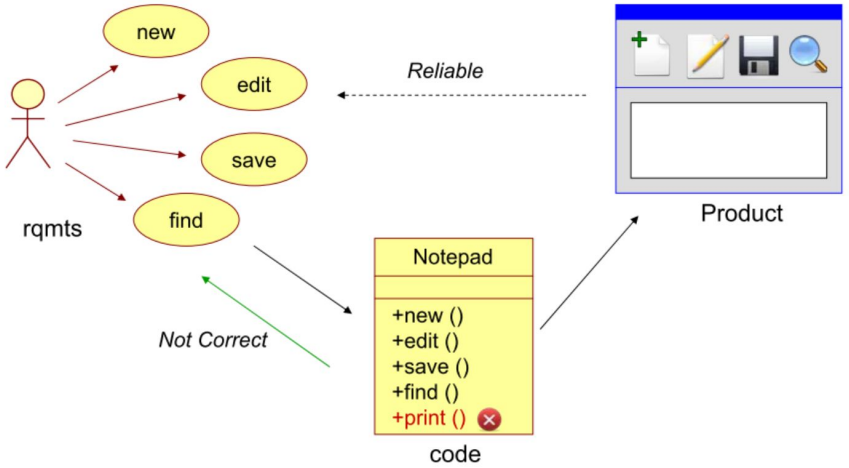
- It is the process used to identify the **correctness**, **completeness** and **quality** of developed computer softwares.
- It is the process of **executing a program/application** under **positive** and **negative conditions** by **manual** or **automated** means. It checks for the:
  - **Specification**
  - **Functionality**
  - **Non-functionality**
  - **Performance**

Software testing - objectives

- **Uncover** as **many** as **errors** (or bugs) as possible in a given product.
- **Demonstrate** a given software product **matching** its requirement **specifications**.
- **Validate** the **quality** of a software testing using the **minimum cost** and efforts.
- Generate **high quality** test cases, perform **effective tests**, and issue correct and **helpful** problem **reports**.

Software testing - Error, Bug, Fault and Failure (1)

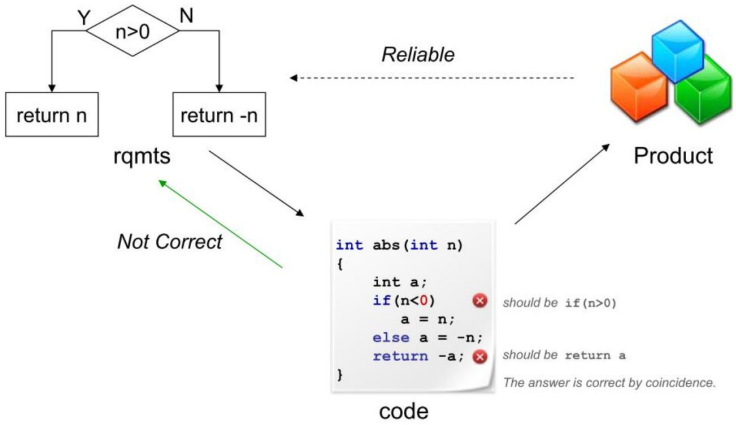
- **Error/mistake**: It is a human action that produces the incorrect result that produces a fault.
  - Syntax/Grammatical error
  - Wrong logic in dealing with customer's requirements.



Example of a redundant implementation

Software testing - Error, Bug, Fault and Failure (2)

- **Error/mistake**: It is a human action that produces the incorrect result that produces a fault.
  - Syntax/Grammatical error
  - Wrong logic in dealing with customer's requirements.



Example of wrong logic

Software testing - Error, Bug, Fault and Failure (3)

- **Bug/defect**: The presence of error at the time of execution of the software. The difference between the actual outcomes and expected outputs.
- **Tracking tools**:
  - Clear Quest
  - DevTrack
  - Jira
  - Quality Center
  - Bugzilla

BUG ID (Autogenerated)	Type	Feature	Description	Priority
BUG-000001	Bug	User Profile	When user uploads a photo, placeholder image doesn't get replaced until the user refreshes. This should happen right away as soon as the upload is completed.	High
BUG-000002	Bug	Marketplace	Marketplace listings are not appearing in reverse chronological order by default as the should be	Critical
BUG-000003	Bug	Registration	Pinch gestures not working in search results pages on Android	Medium
BUG-000004	Investigation	Billing	Credit card approval notifications delayed by longer than expected	Critical - High Priority
BUG-000005	Regression	Page Editor	Line breaks lost in embedded code	High
BUG-000006	Bug	User Profile	Settings option shows as highlighted even when mouse is not hovered over it	Medium

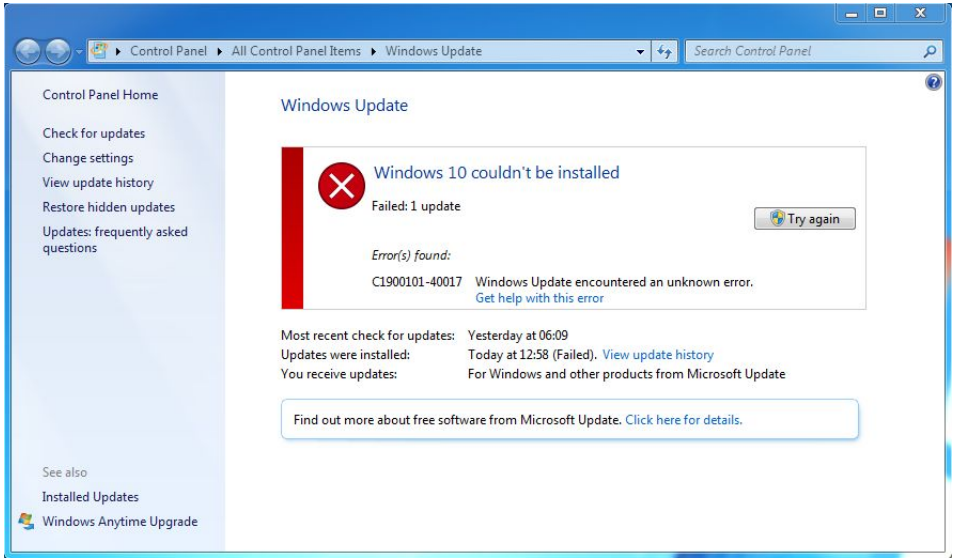
Defect tracking report

Software testing - Error, Bug, Fault and Failure (4)

- **Fault**: State of software caused by an error causes the software to fail to accomplish its essential function.

Software testing - Error, Bug, Fault and Failure (5)

- **Failure**: Deviation of the software from its expected result. It is an event. If the software has lots of defects, it leads to failure or causes failure.

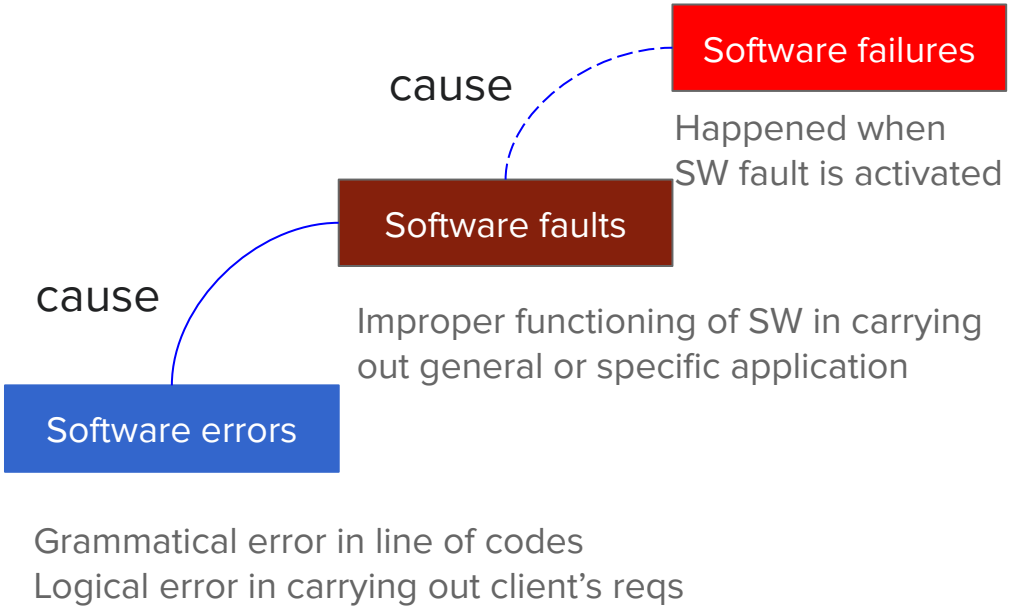


Example of a failure

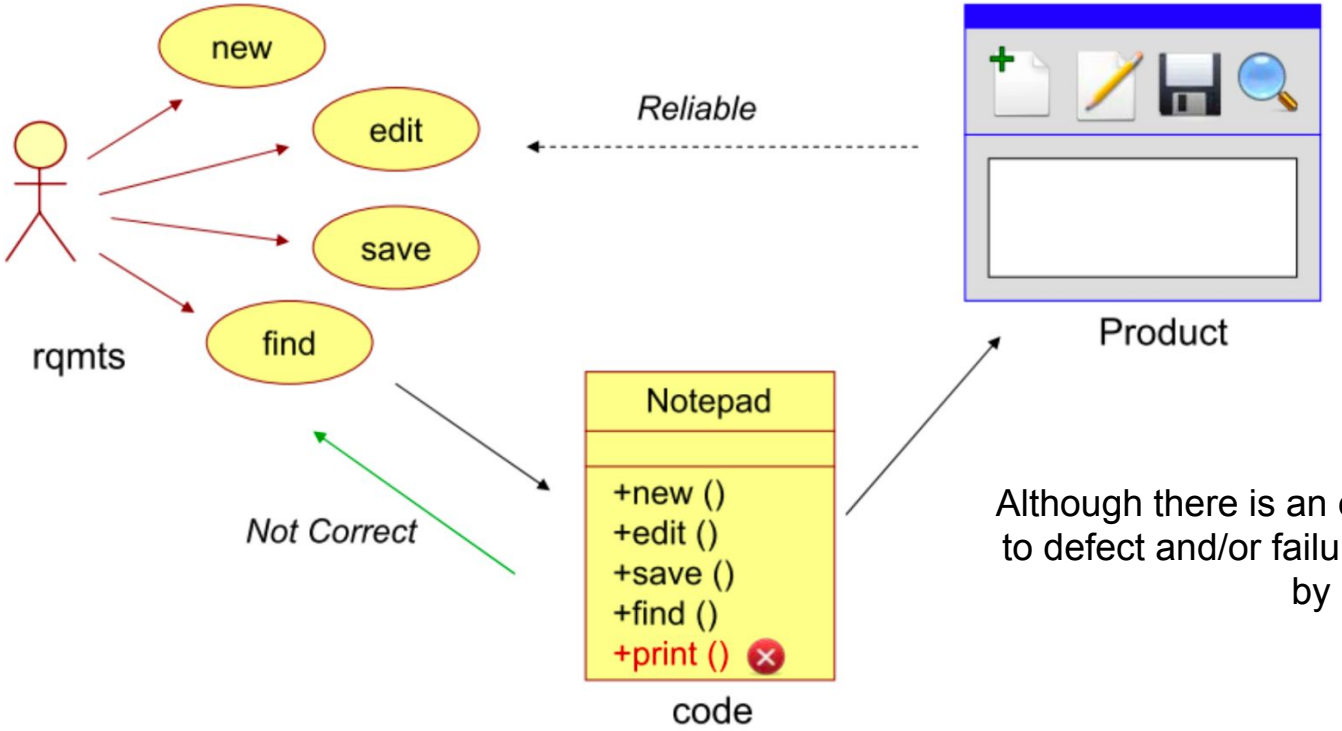
Bug/fault/failure relation

Nevertheless, in some cases:

- **Errors may not cause defects** because errors are **fixed** or **deactivated** by other logics/lines of code.
- **Defects** may not **cause failures** when there is no impact on user yet or not sufficient condition to occur.

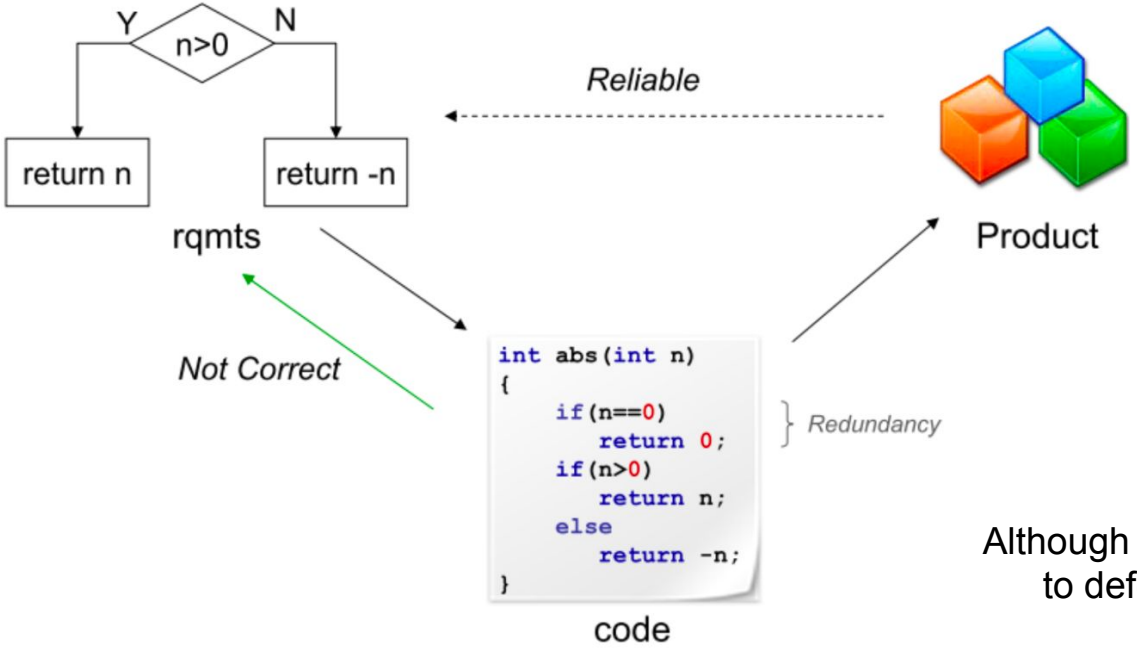


Bug/fault/failure relation example (1)



Although there is an error, this does not lead to defect and/or failure as “print” is not used by users

Bug/fault/failure relation example (2)



Although there is an error, this does not lead to defect and/or failure as the code is deactivated by next ones.

Bug/fault/failure causes (1)

Software errors are the **cause** of **poor** software quality that can be

- **Code Error**
- **Procedure Error**
- **Documentation Error**
- **Software Data Error**

Bug/fault/failure causes (2)

The 9 Causes of Software Errors

1. Faulty requirement definition
2. Client-developer communication failures
3. Deliberate deviation form software requirements
4. Logical design errors
5. Coding errors
6. Non-compliance with documentation and coding
7. Shortcomings of the testing process
8. Procedure errors
9. Documentation errors

Bug/fault/failure summary (1)

Basis	Bug	Defect	Error	Fault	Failure
Definition	It is an informal name specified to the defect.	The **Defect** is the difference between the actual outcomes and expected outputs.	An **Error** is a mistake made in the code; that's why we cannot execute or compile code.	The **Fault** is a state that causes the software to fail to accomplish its essential function.	If the software has lots of defects, it leads to failure or causes failure.

Bug/fault/failure summary (2)

Basis	Bug	Defect	Error	Fault	Failure
Raised by	The **Test Engineers** submit the bug.	The **Testers** identify the defect. And it was also solved by the developer in the development phase or stage.	The **Developers and automation test engineers** raise the error.	**Human mistakes** cause fault.	The failure finds by the manual test engineer through the **development cycle**.

Bug/fault/failure summary (3)

Basis	Bug	Defect	Error	Fault	Failure
Different types	Different type of bugs are as follows: ○ Logic bugs ○ Algorithmic bugs ○ Resource bugs	Different type of Defects are as follows: Based on **priority**: ○ High ○ Medium ○ Low And based on the severity: ○ Critical ○ Major ○ Minor ○ Trivial	Different type of Error is as below: ○ Syntactic Error ○ User interface error ○ Flow control error ○ Error handling error ○ Calculation error ○ Hardware error ○ Testing Error	Different type of Fault are as follows: ○ Business Logic Faults ○ Functional and Logical Faults ○ Faulty GUI ○ Performance Faults ○ Security Faults ○ Software/ hardware fault	-----

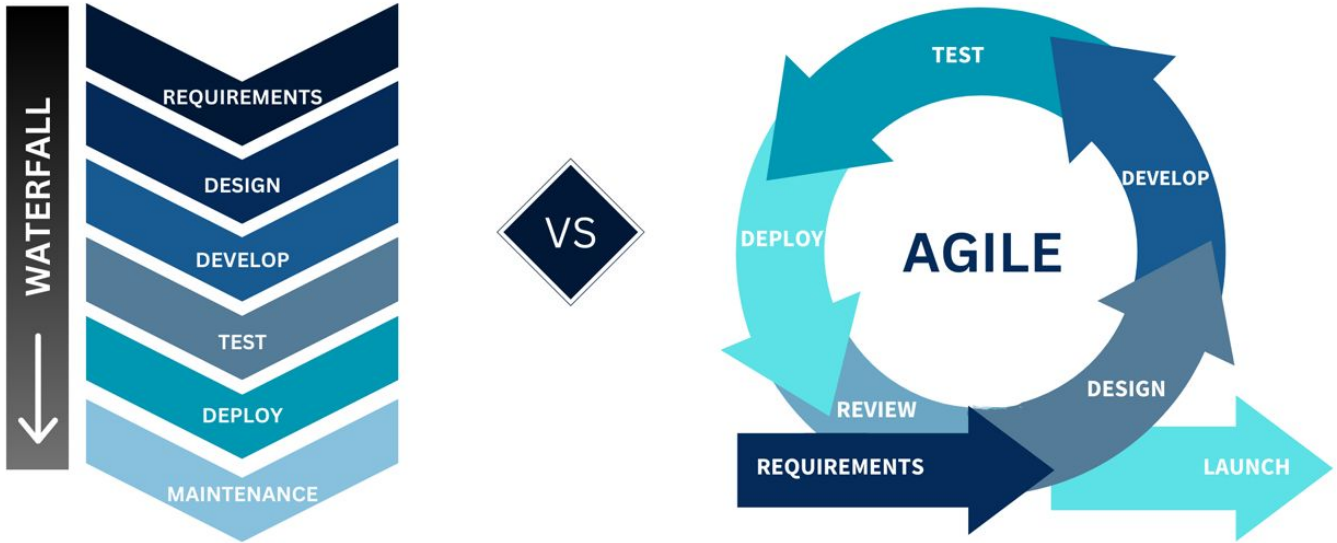
Bug/fault/failure summary (4)

Basis	Bug	Defect	Error	Fault	Failure
Reasons behind	Following are reasons which may cause the **bugs:** Missing coding Wrong coding Extra coding	The below reason leads to the **defects**: Giving incorrect and wrong inputs. Dilemmas and errors in the outside behavior and inside structure and design. An error in coding or logic affects the software and causes it to breakdown or the failure.	The reasons for having an **error** are as follows: Errors in the code. The Mistake of some values. If a developer is unable **to compile or run a program successfully.** Confusions and issues in programming. Invalid login, loop, and syntax. Inconsistency between actual and expected outcomes. Blunders in design or requirement actions. Misperception in understanding the requirements of the application.	The reasons behind the **fault** are as follows: A Fault may occur by an improper step in the initial stage, process, or data definition. Inconsistency or issue in the program. An irregularity or loophole in the software that leads the software to perform improperly.	Following are some of the most important reasons behind the **failure:** Environmental condition System usage Users Human error

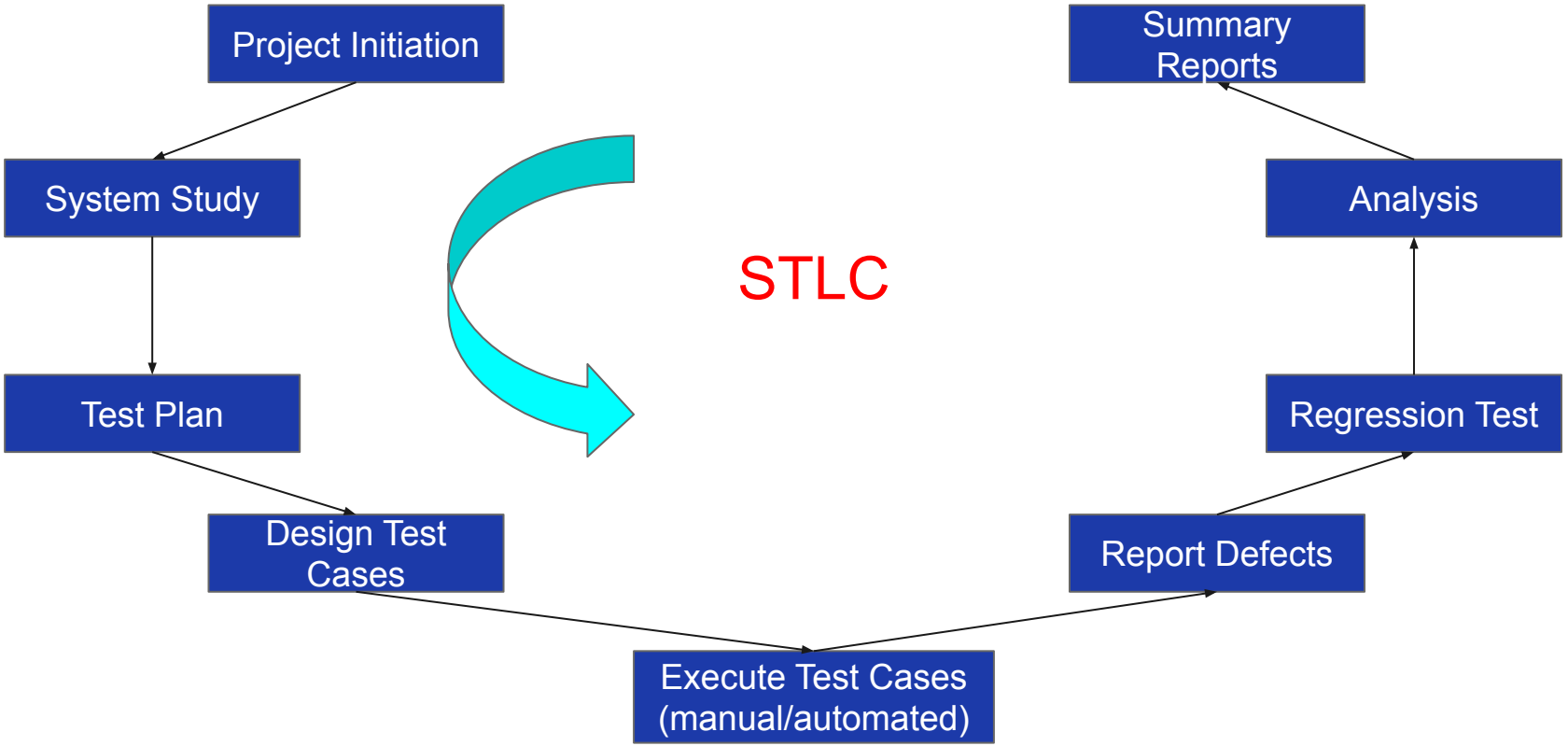
Bug/fault/failure summary (4)

Basis	Bug	Defect	Error	Fault	Failure
Way to prevent the reasons	Following are the way to stop the **bugs**: Test-driven development. Offer programming language support. Adjusting, advanced, and operative development procedures. Evaluating the code systematically.	With the help of the following, we can prevent the **Defects**: Implementing several innovative programming methods. Use of primary and correct software development techniques. Peer review It is executing consistent code reviews to evaluate its quality and correctness.	Below are ways to prevent the **Errors**: Enhance the software quality with system review and programming. Detect the issues and prepare a suitable mitigation plan. Validate the fixes and verify their quality and precision.	The **fault** can be prevented with the help of the following: Peer review. Assess the functional necessities of the software. Execute the detailed code analysis. Verify the correctness of software design and programming.	The way to prevent **failure** are as follows: Confirm re-testing. Review the requirements and revisit the specifications. Implement current protective techniques. Categorize and evaluate errors and issues.

Waterfall vs Agile Model



Testing Life Cycle



Test Plan

- Test plan is a **systematic approach** to test a system e.g software.
- The plan typically **contains** a **detailed understanding** of what the eventual **testing workflow** will be.

Test case

- Test case is **a specific procedure** of testing a particular requirement.
- Test case will **include**:
  - **Identification** of specific **requirements** tested.
  - Test case **success/failure criteria.**
  - Specific **steps** to execute test.
  - Test **data**.

Verification vs Validation

- **Verification**: The software should confirm to its specifications

  (Are developers building the product right?)

- **Validation**: The software should do what the user really requires

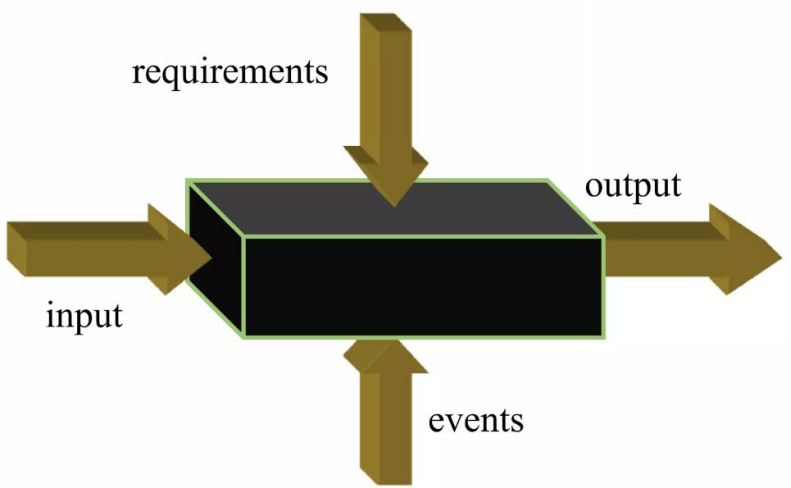
  (Are developers building the right product?)

Testing Methodologies

- **Black box testing**
- **White box testing**

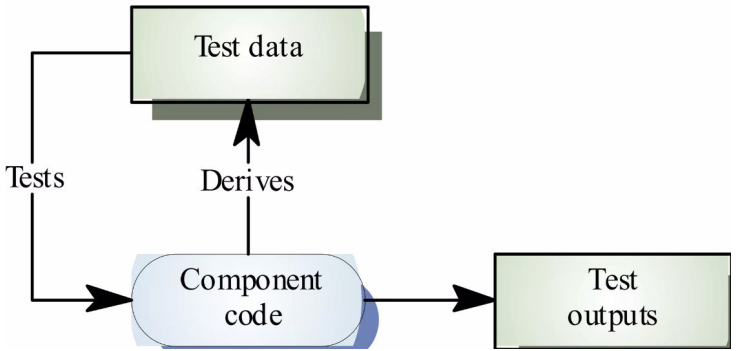
Testing Methodologies

- **Black box testing**
  - **No knowledge** of internal program design or code required.
  - Tests are **based on requirements and functionality.**



Testing Methodologies

- **White box testing**
  - **Knowledge** of the internal program design and code required.
  - Tests are **based on coverage of code statements, branches, paths, conditions.**



Testing Levels

- **Unit testing**
- **Integration testing**
- **System testing**
- Acceptance testing

Unit testing

- Test each module **individually**
- Follow a **white box testing** (logic of the program)
- Done by **developers**

Integration testing

- Once all the modules have been **unit tested**, integration testing is performed.
- It is **systematic testing**.
- Produce tests to identify errors associated with interfacing
- Types:
  - Big Bang Integration testing
  - Top Down Integration testing
  - Bottom Up Integration testing
  - Mixed Integration testing

System testing (1)

- The system as a whole is tested to **uncover requirement errors.**
- Verifies that all **system** elements **work properly** and that overall system **function** and **performance** has been **achieved**.
- Types:
  - Alpha Testing
  - Beta Testing
  - Acceptance Testing
  - Performance Testing

System testing (2)

- **Alpha Testing**:
  Carried out by the test team within the developing organization.
- **Beta Testing**:
  Performed by a selected group of friendly customers
- **Acceptance Testing**:
  Performed by the customer to determine whether to accept or reject the delivery of the system
- **Performance Testing:**
  Carried out to check whether the system meets the non-functional requirements identified in the document.

System testing - types of performance testing

- Stress Testing
- Volume Testing
- Configuration Testing
- Compatibility Testing
- Regression Testing
- Recovery Testing
- Maintenance Testing
- Documentation Testing
- Usability Testing

Software Testing Discussion

- In order to be cost effective, the testing must be focused on areas where it will be most effective
- The testing should be planned such that when testing is stopped for whatever reason, the most effective testing in the time allotted has already been done
- The absence of an organizational testing policy may result in too much effort and money will be spent on testing, attempting to achieve a level of quality that is impossible or unnecessary

References

[1]. Kshirasagar Naik, Priyadarshi Tripathy, [2008], Software Testing and Quality Assurance, John Wiley & Sons, New Jersey.

[2]. Mauro Pezze, Michal Young, [2008], Software Testing and Analysis: Process, Principles, and Techniques, John Wiley & Sons, New Jersey.

[3]. Arnon Axelrod, [2018], Complete Guide to Test Automation: Techniques, Practices, and Patterns for Building and Maintaining Effective Software Projects, Matan, Israel.

[4] Trần Cao Đệ, Nguyễn Công Danh, Giáo trình đảm bảo chất lượng phần mềm, NXB. Đại học Cần Thơ, 2014.