ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

Software Testing

Chapter 2
Testing in the Software Developement Life Cycle (SDLC)

http://vku.udn.vn/

Contents

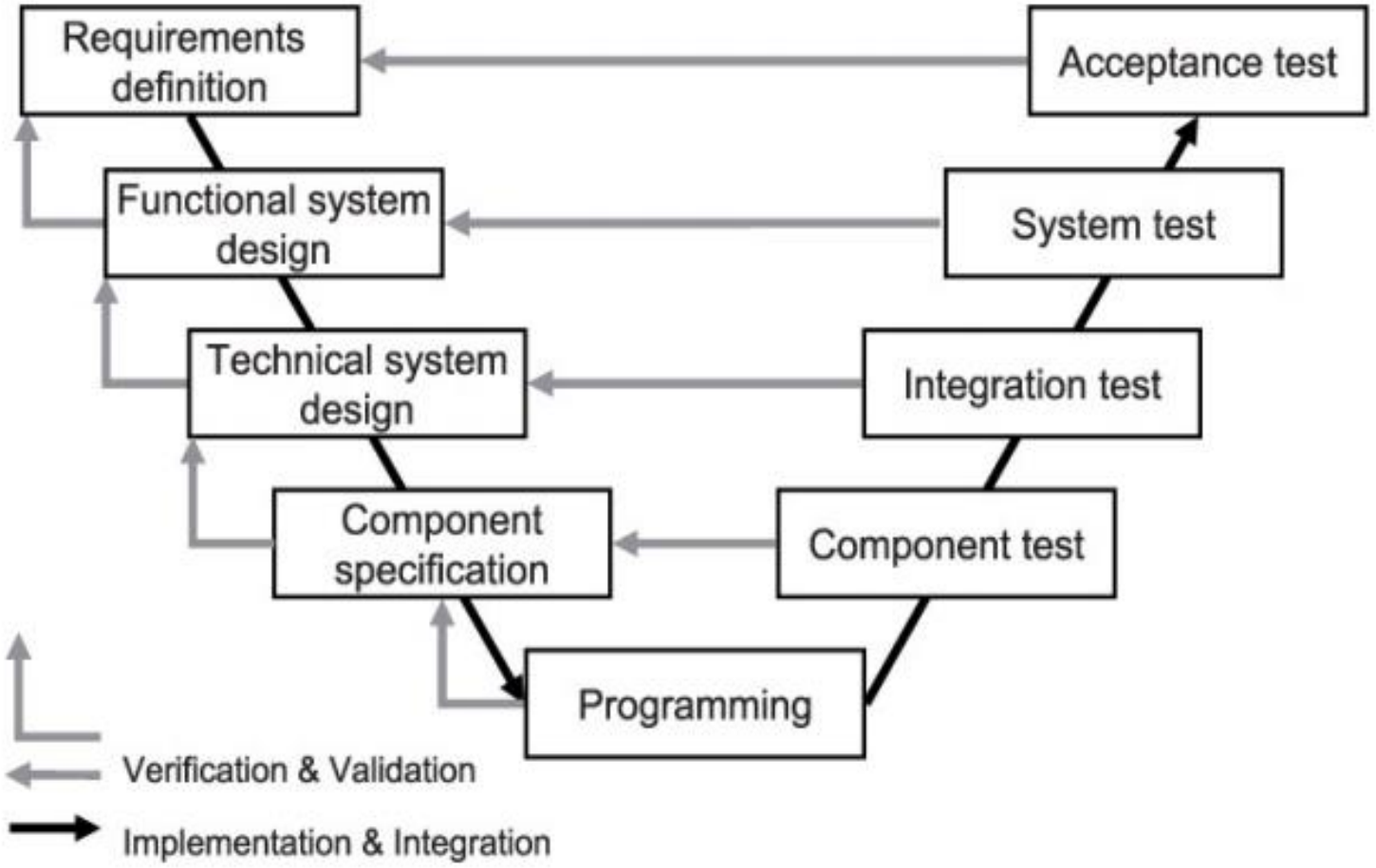
1. The Role of Software Testing in SDLC
2. Definitions: Verification, Validation, QA, QC
3. Test Levels
4. Types of Testing

1.The Role of Software Testing in SDLC

- Software development models:
  - A development life cycle for a software product involves capturing the initial requirements from the customer, expanding on these to provide the detail required for code production, writing the code and testing the product, ready for release.
- There are two common models:
  - Sequential development models: Waterfall, V-model
  - Iterative and incremental development models: Scrum, Spiral, Agile, Kanban, Rational Unified Process…

V-Model

- The two branches of the V symbolize this.



V-Model

- The left branch represents the development process:
  - Requirements definition
    - ✓capturing of user needs
  - Functional specification
    - ✓Definition of functions required to meet user needs
  - Technical system design
    - ✓technical design of functions identified in the functional specification
  - Component specification
    - ✓detailed design of each module or unit to be built to meet required functionality
  - Programming: use programming language to build the specified component (module, unit, class…)

V-Model

- The right branch defines a test level for each specification and construction level
  - Component test:
    - ✓Verifies whether each software, component correctly fulfills its specification.
  - Integration test
    - ✓Checks if groups of components interact in the way that is specified by the technical system design.
  - System test
    - ✓Verifies whether the system as a whole meets the specified requirements.
  - Acceptance test
    - ✓Checks if the system meets the customer requirements, as specified in the contract and/or if the system meets user needs and expectations.

V-Model

- Some most important characteristics behind the V-model:
  - For every development activity, there is a corresponding testing activity.
  - Each test level has test objectives specific to that level
  - Test analysis and design for a given test level begin during the corresponding development activity
  - Tester should be involved in reviewing documents as soon as drafts are available in the development life cycle.
  - V-model illustrates the testing aspects of verification and validation.

Verification

Are we building the product **right**?

To ensure that work products meet their specified requirements.

Validation

Are we building the **right** product?

To ensure that the product actually meets the user's needs, and that the specifications were correct in the first place.

2.2 Quality Assurance vs Quality Control

❖Quality Control:

- Is a process that focuses on fulfilling the quality requested.
- Aims to **identify and fix defects**
- QC involves in full software testing life cycle
- QC activities are only a part of the total range of QA activities.

❖Quality Assurance:

- It is a process that focuses on providing assurance that quality request will be achieved.
- Aims to **prevent causes of detect** and correct them early in the development process
- QA involves in full software development life cycle

3. Test Levels

- Test level (Test stage) is a group of test activities that are organized and managed together. A test level is linked to the responsibilities in a project.
- Test level consists of:
  - Component Testing
  - Integration Testing
  - System Testing
  - Acceptance Testing
  - Regression Testing

3.1 Component Testing

- Is also known as unit testing, module testing, program testing
- Is the testing of individual program units, such as a procedures, functions, methods, or classes, in *isolation*.
- The goal of Unit testing is to *ensure that the code written for the unit meets its specification*, prior to its integration with other units.
- Testing
  - Functionality
  - Non- functional characteristics

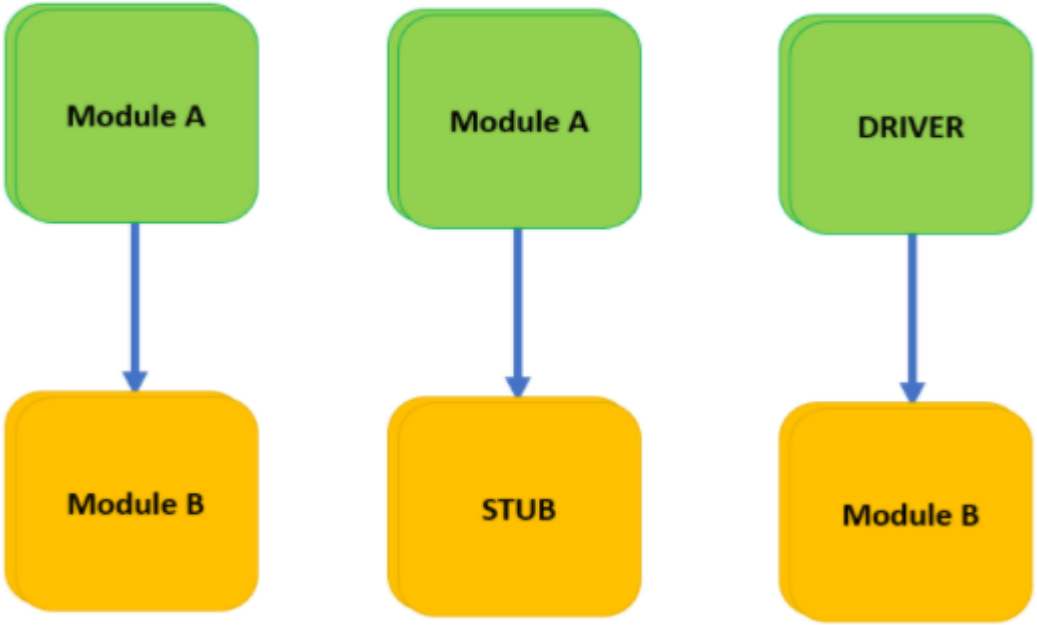
3.1 Component Testing

- Usually component testing is performed by the developer
- In test-driven development (TDD), which relates to Agile Software Development model, unit tests are prepared by developers *before code was written*.
- As unit testing is made in isolation from the rest of the system, there can be missing parts of software

=>Stubs and Drivers are used to replace the missing software and simulate the interface between the software components in a simple manner.

3.1 Component Testing

- A stub is called from the software component to be tested; a driver calls a component to be tested
- Stub and driver:



3.1 Component Testing

❖Stub and driver:

Stubs	Drivers
Stubs are used in top down testing approach	Drivers are used in bottom up testing approach
Stubs are used when the major module is ready to test, but the sub modules are not ready yet	Drivers are used when the sub modules are ready, but the main module is not ready yet
Stubs are "called" programs	Drivers are "calling" programs

3.2 Integration testing

- Once the units have been written, the next stage is to put them together to create the system. This is called integration. It involves building something large from a number of smaller pieces.
- Integration testing *tests interfaces between components*, *interactions to different parts of a system* such as an operating system, file system and hardware or *interfaces between systems*.

3.2 Integration testing

❖Levels of integration testing:

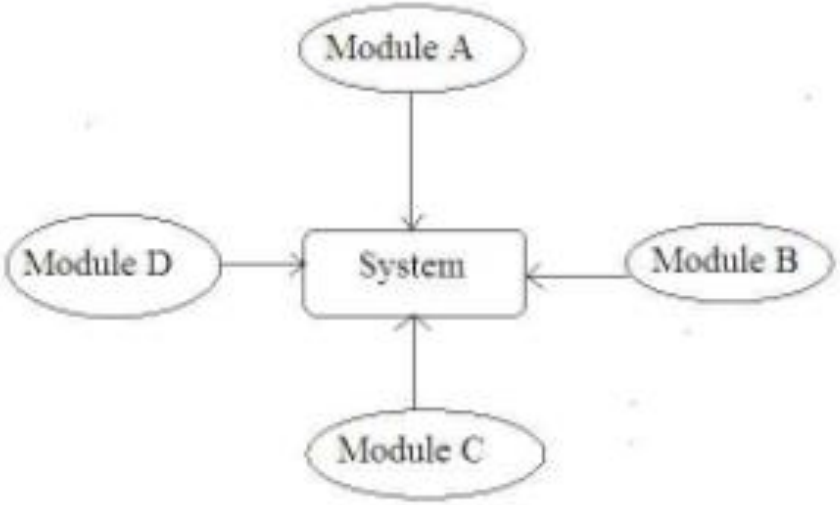
- **Component integration testing**
  - ✓Verifies the interactions between software components and is done after component testing.
  - ✓is usually carried out by **developers**.
- **System integration testing**
  - ✓Verifies the interactions between different systems and may be done after system testing each individual system.
  - ✓For example, a trading system in an investment bank will interact with the stock exchange to get the latest prices for its stocks and shares on the international market.
  - ✓This type of integration testing is usually carried out by **testers**.

3.2 Integration testing

- There are three commonly integration strategies:
  - Big-bang integration
  - Top-down integration
  - Bottom-up integration

3.2 Integration testing

- Big-bang integration:
  - testing all components or modules are integrated simultaneously, after which everything is tested as a whole.
  - Individual modules are not integrated until and unless all the modules are ready.



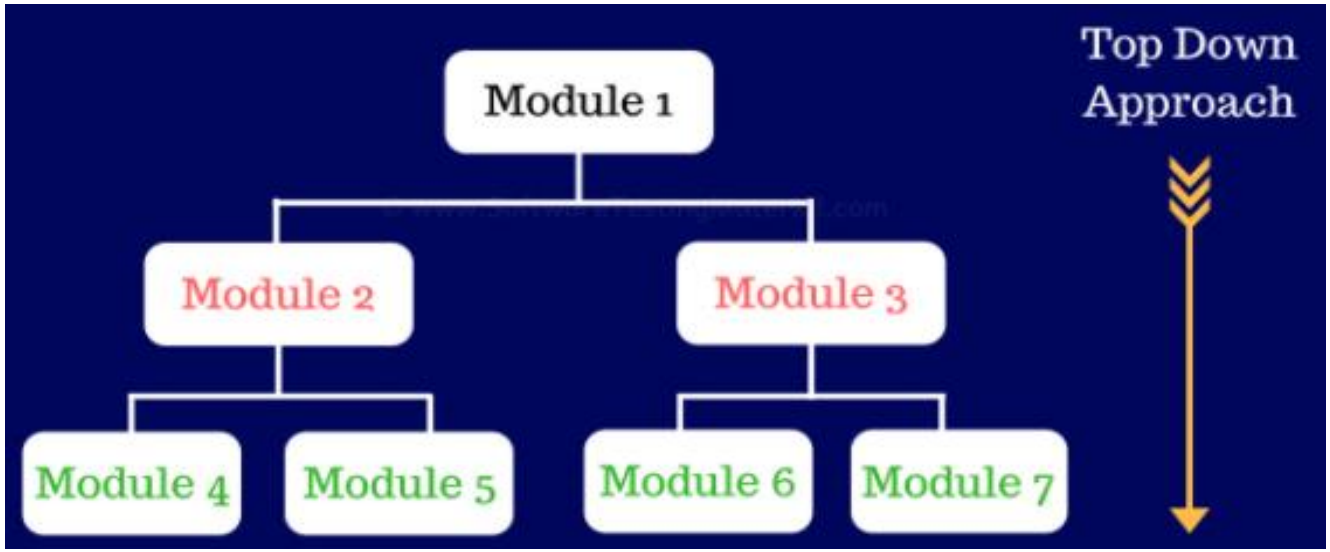
3.2 Integration testing

- Big-bang integration:
  - Advantage: everything is finished before integration testing starts.
  - Disadvantages:
    - ✓in general it is very time consuming
    - ✓It is very difficult to trace the cause of failures because of this late integration.
    - ✓The chances of having critical failures are more because of integrating all the components together at same time.
    - ✓If any bug is found then it is very difficult to detach all the modules in order to find out the root cause of it.
    - ✓There is high probability of occurrence of the critical bugs in the production environment

3.2 Integration testing

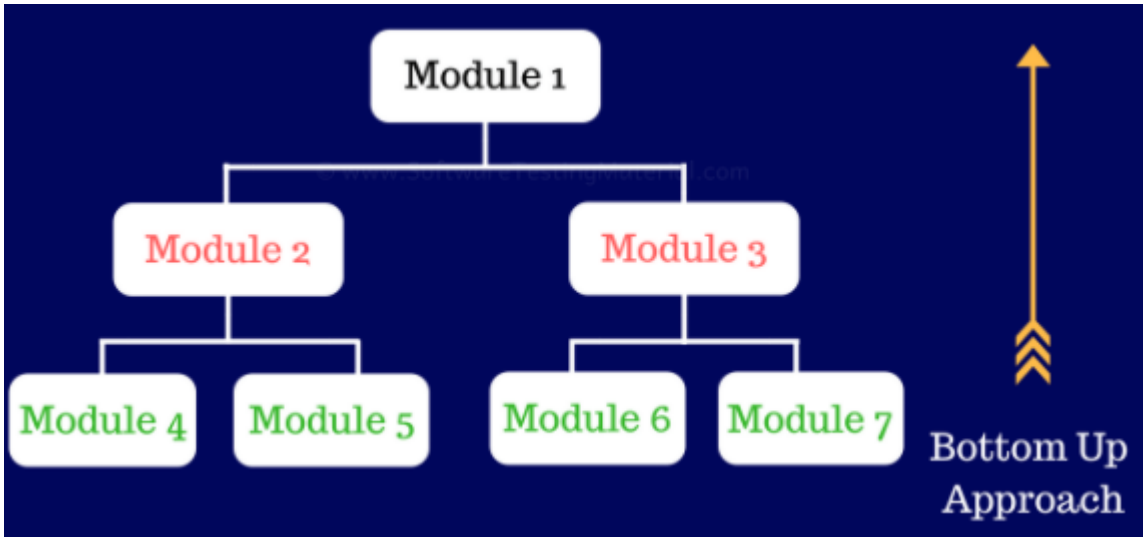
❖Top-down integration:

- Testing takes place from top to bottom. High-level modules are tested first and then low-level modules and finally integrating the low-level modules to a high level to ensure the system is working as intended.
- Stubs are used as temporary module if a module is not ready for integration testing.



3.2 Integration testing

- Bottom-up integration:
  - Testing takes place from bottom to up. Lowest level modules are tested first and then high-level modules and finally integrating the high-level modules to a low level to ensure the system is working as intended.
  - Drivers are used as a temporary module for integration testing.



3.3. System testing

- System testing is testing an integrated system to verify that it meets specified requirements.
- It may include tests based on risks and/or requirement specifications, business process, use cases, or other high level descriptions of system behavior, interactions with the operating systems, and system resources.
- System testing is carried out by specialists testers or independent testers.

3.3. System testing

- System testing:
  - non-functional tests include performance and reliability. Testers may also need to deal with incomplete or undocumented requirements.
  - functional requirements starts by using the most appropriate specification-based (black-box) techniques
- System testing requires a product environment

3.4 Acceptance testing

- Acceptance testing is software testing performed on software prior to its delivery.
- The goal of acceptance testing is not to find defects, but to provide the end users with confidence that the system will function according to their expectations and can be released.
- How much acceptance testing?
  - Depend on the product risk.

3.4 Acceptance testing

- There are four typical forms of acceptance testing:
  - Contract and regulation acceptance testing
  - User acceptance testing (UAT)
  - Operational acceptance testing (OAT)
  - Alpha and beta testing

3.4 Acceptance testing

- Alpha and beta testing are two stages of acceptance testing
  - Alpha testing:
    - ✓Is testing by a potential users or an independent test team *at the developing organization's site* before the software product is released to external customers.
    - ✓It is a form of internal acceptance testing.
  - Beta testing:
    - ✓Is testing by a group of customers/potential users who use the product at their own locations and provide feedback *at the customer's site* , before the software product is released.
    - ✓ It is a form of external acceptance testing.

3.4 Acceptance testing

- Alpha and beta testing:
  - are usually performed on commercial off-the-shelf software (COTS). It is software that was developed for the mass market.
  - Done to get feedback from potential or existing users/customer before the software product is released to the market
  - Alpha testing is done before Beta testing

4. Test type

- Test Type is a group of test activities aimed at testing a component or system focused on a specific test objective.
- Test types:
  - Functional testing.
  - Non-functional testing.
  - Structural testing.
  - Testing related to changes (Retesting and Regression testing)

4.1 Functional testing

- Functional testing is testing of functions.
  - Objective: test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Testing what the system does

- Functional testing is also called **specification-based testing**

4.1 Functional testing

- The functionality of the application is usually described in the following documents:
  - Functional specifications
  - Requirements specifications
  - Business requirements
  - Use cases
  - User stories…
- Functional testing mainly involves black box testing and it is not concerned about the source code of the application

4.1 Functional testing

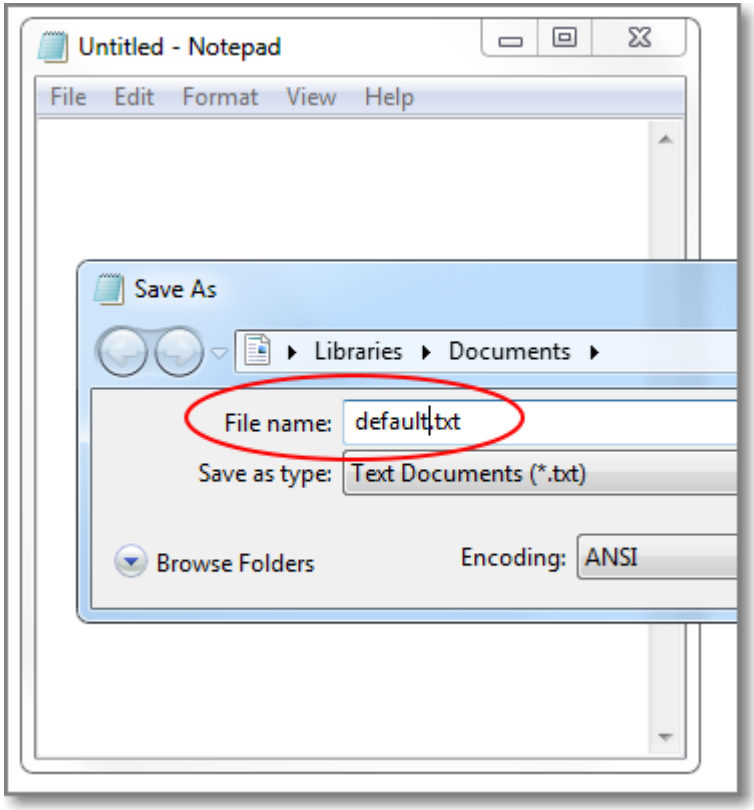
- Functional testing focuses on:
  - Security
  - Suitability
  - Accuracy
  - Interoperability (compatibility)
  - compliance
- Example:
  - User can login the website successfully with valid account
  - User is able pay for the purchase in the e-store using Visa
  - Firewall can detect threats such as virus

4.1 Functional testing example

Task: Test Save feature of Notepad application.

Functional Testing Procedure: test different flows of Save functionality (Save new file, save updated file, test Save As, save to protected folder, save with incorrect name, re-write existed document, cancel saving, etc.)

Defect: While trying to save file using Save As command, still default file name can only be used. User cannot change the filename because the edit-box is disabled.



4.2 Non-Functional testing

- Non-Functional testing is testing of Non-functional software characteristics

Testing of how the system works

- Objective: a non-functional characteristics that do not relate to functionality of a software application.
- Non-functional characteristics are:
  - Reliability
  - Usability
  - Efficiency
  - Maintainability
  - Portability

4.2 Non-Functional testing

- Types of Non-Functional testing:
  1. Performance testing
     - ✓Load testing
     - ✓Stress testing
     - ✓Endurance testing
     - ✓Volume testing
     - ✓Spike Testing
  2. Document Testing
  3. Installation Testing
  4. Reliability Testing
  5. Security Testing

4.2.1 Performance testing

- Performance testing is a type of performance testing to determine how efficiently a product handles a variety of events.
  - Ex: Measuring the response time of a website, or a specific element
- Tester needs to check product's speed, scalability and stability
  - Speed : Need to verify that application response is getting quick or not.
  - Scalability : Need to verify that application can handle maximum user load.
  - Stability : Need to verify that application is stable under varying load.

4.2.1 Performance testing: Example

Task: Server should respond in less than 2 sec when up to 100 users access it concurrently. Server should respond in less than 5 sec when up to 300 users access it concurrently.

Performance Testing Procedure: emulate different amount of requests to server in range (0; 300), for instance, measure time for 10, 50, 100, 240 and 290 concurrent users.

Defect: starting from 200
Concurrent requests
respond time is 10-15 seconds.

4.2.2.1 Load testing

- Load testing is testing which involves evaluating the performance of the system under the *expected workload* to determine what load can be handled by the component or system.
- Expected workload: numbers of parallel users, numbers of transactions…
- Example:
  - Uploading/downloading big volume files
  - Ordering item in online store by a big amount of users simultaneously

4.2.2.1 Load testing: Example

Task: Server should allow up to 500 concurrent connections.

Load Testing Procedure: emulate different amount of requests to server close to pick value, for instance, measure time for 400, 450, 500 concurrent users.

Defect: Server returns "Request Time Out" starting from 490 concurrent requests.

4.2.2.2 Stress testing

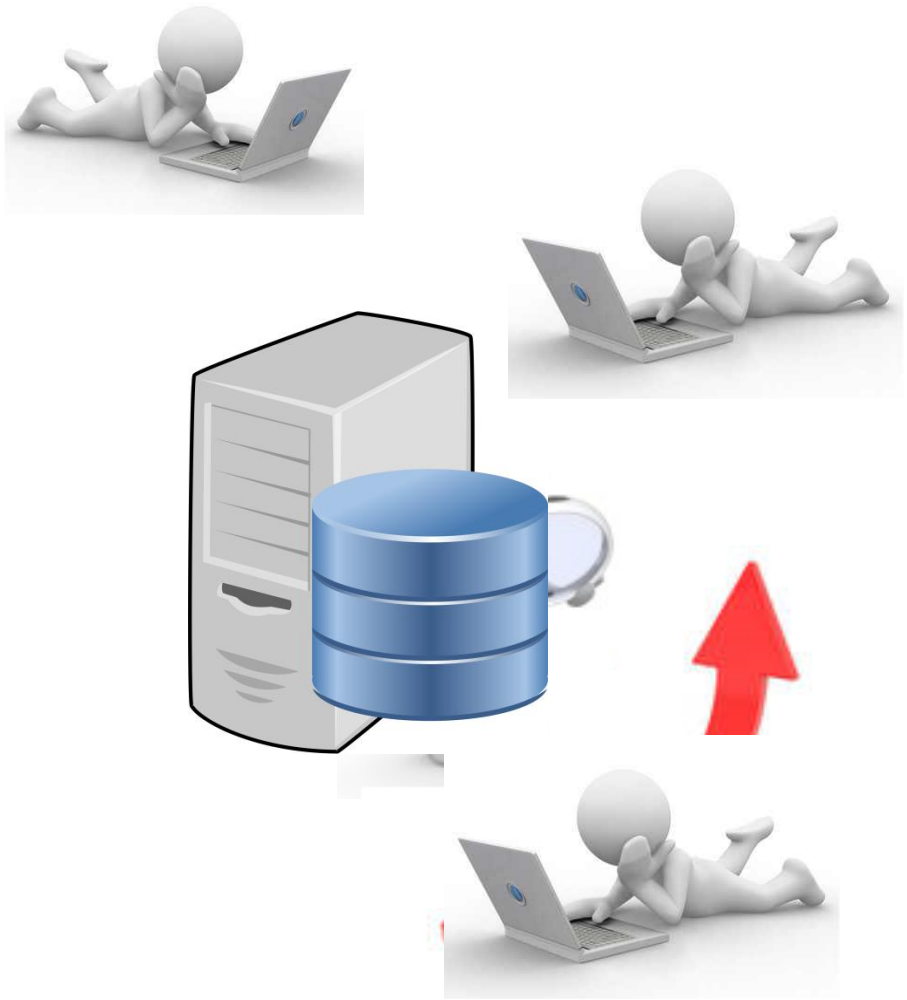
- Stress testing: is a type of performance testing conducted to evaluate a system or component at or beyond the limits of its anticipated or specified work loads, or with reduced availability of resources such as access to memory or servers

4.2.2.2 Stress testing: Example

Task: Server should allow up to 500 concurrent connections.

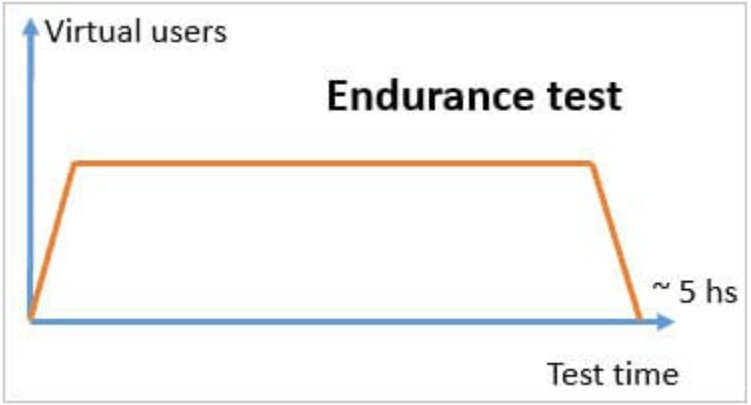
Stress Testing Procedure: emulate amount of requests to server greater than pick value, for instance, check system behavior for 500, 510, and 550 concurrent users.

Defect: Server crashes starting from 500 concurrent requests and user's data is lost.
Data should not be lost even in stress situations. If possible, system crash also should be avoided.



4.2.2.3 Endurance testing

- Endurance testing is a testing of the software to check system performance under specific load conditions over an extended or longer amount of time.
- It is also known as Soak testing



4.2.2.3 Endurance testing: Example

- A system may behave exactly as expected when tested for 1 hour but when the same system is tested for 3 hours, problems such as memory leaks cause the system to fail or behave randomly.
- For an application like Income tax filing, the application is used continuously for a very long duration by different users. In this type of application, memory management is very critical.

4.2.2.4 Volume testing

- Volume testing refers to testing a software application or the product with a certain amount of data.
- E.g., if we want to volume test our application with a specific database size, we need to expand our database to that size and then test the application's performance on it.

Type of testing in which the system is tested on large data volumes

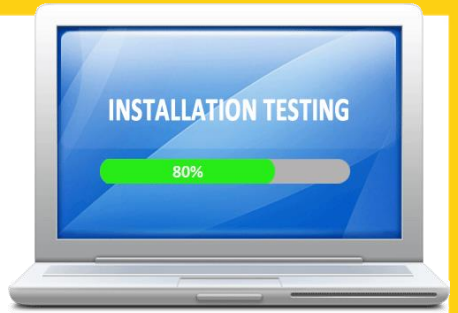
4.2.2.5 Spike Testing

- It refers to test conducted by subjecting the system to a short burst of concurrent load to the system. This test might be essential while conducting performance tests on an auction site wherein a sudden load is expected.
  - **Example** – For an e-commerce application running an advertisement campaign, the number of users can increase suddenly in a very short duration.

4.2.2. Document Testing

- Document Testing means verify the technical accuracy and readability of the user manuals, tutorials and the online help.
- Document testing:
  - a spelling and grammar checking in the documents using available tools
  - to manually reviewing the documentation to remove any ambiguity or inconsistency.
- Can start at the very beginning of the oftware process and hence save large amounts of money

4.2.2.3. Installation Testing



- Installation testing is performed to check if the *software has been correctly installed* with all the inherent features and that the product is working as per expectations.
- Also known as **implementation testing**
- It is done in the last phase of testing before the end user has his/her first interaction with the product.

4.2.2.4.Reliability Testing

- Reliability testing is performed to ensure that the software is reliable, it satisfies the purpose for which it is made, for a specified amount of time in a given environment and is capable of rendering a fault-free operation

4.2.2.5. Security Testing

- Security testing is a testing technique to determine if an information system protects data and maintains functionality as intended.
- Example:
  - A password should be in encrypted format
  - Application or System should not allow invalid users
  - Check cookies and session time for application

4.3 Structural testing

- The structural testing is the testing of the structure of the system or component.

Testing through assessment of coverage of a type of structure

- Structural testing is often referred to as ‘white box’ testing because in structural testing we are interested in what is happening ‘inside the system/application’.

4.3 Structural testing

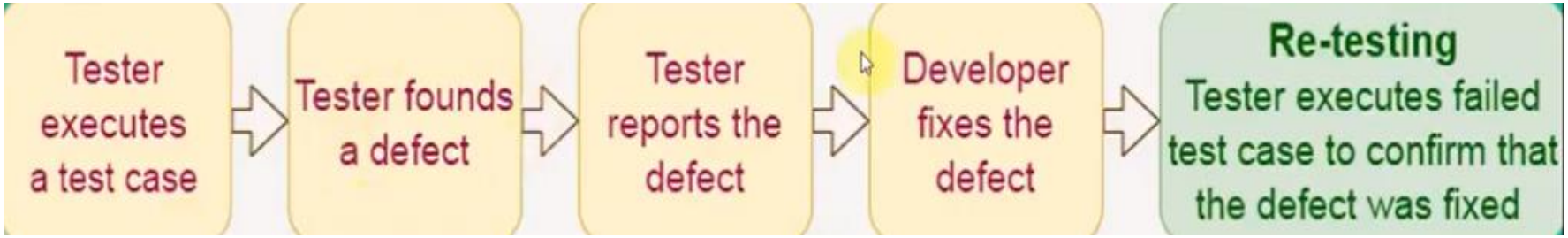
- Structural testing can be used at all levels of testing
  - Developers use structural testing in component testing and component integration testing, especially where there is good tool support for **code coverage**.
- Coverage measurement tools assess the percentage of executable elements (e.g. statements or decision outcomes) that have been exercised (i.e. covered) by a test suite.

4.4 Testing related to changes

- Objective: changes related to defects that have been fixed and looking for unintended changes
- When a defect has been corrected, two types of tests should be executed:
  - Confirmation testing (retesting)
  - Regression testing

4.4 Testing related to changes

- **Confirmation testing(Re-testing)** is testing a defect again – after it has been fixed.
- When developers have fixed the defect, it is assigned for testers to re-test test cases with the same inputs, data and environment to **confirm** the defect was really fixed.



4.4 Testing related to changes

- **Regression testing** is a testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in **unchanged areas** of the software, as a result of the changes made.
- Changes can be made in the software or environment.
- New defect can appear when
  - Software was changed: adding/fixing functionality, refactoring…
  - Environment was changed

4.4 Testing related to changes

Regression Testing	Re-Testing
In Regression testing, you can include the test cases which passed earlier. We can say that check the functionality which was working earlier.	In Retesting, you can include the test cases which failed earlier. We can say that check the functionality which was failed in earlier build.
Regression testing can done by Automation testing	Retesting cant be done by Automation testcases
Defect verification is not the part of Regression Testing	Defect verification is the part of re-testing
Regression testing checks for unexpected side-effects	Re-testing makes sure that the original fault has been corrected
Regression testing is only done when there is any modification or changes become mandatory in an existing project	Re-testing executes a defect with the same data and the same environment with different inputs with a new build

Regression testing = find new bugs.
Retesting = confirm bugs were fixed.
Regression testing locates bugs after updates or changes to the code or UI.
Retesting makes sure that a bug that was found and addressed by a developer was *actually* fixed

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

Thank You !