

Test Approaches

Test Levels

Test Types

June 2017, 2018

softserve

Agenda

- Test Approaches
- Test Types by Test Levels
- Test Types by Test Objectives
- Testing Order



Test Types

Acceptance
Performance
Stress
Compatibility
Recovery
Regression
Localization

Security
Load
System
Stress
Functional
Exploratory
Smoke
Confirmation

Black Box
Grey Box
GUI
Unit
Install/uninstall

Usability
White Box
Sanity
Error-Handling
Negative

Conformance
Internationalization
Beta
Ad-hoc
Integration
Capacity
Alpha

Test Type Definition

Test Type it's a group of test activities aimed at testing a component or system focused on a specific test objective

Test activities can be grouped by:

- Test Approaches
- Test Levels
- Test Objectives



Test Approaches

Test Approaches

- Proactive and Reactive
- Manual and Automated
- Verification and Validation
- Positive and Negative
- Black-box, White-box and Grey-box
- Scripted and Unscripted



softserve

Proactive and Reactive testing

Reactive behavior is reacting to problems when they occur instead of doing something to prevent them



Testing is not started until design and coding are completed

Proactive behavior involves acting in advance of a future situation, rather than just reacting.




Test design process is initiated as early as possible in order to find and fix the defects before the build is created



Manual and Automated

Manual testing is the process through which software developers run tests manually, comparing program expectations and actual outcomes in order to find software defects

VS

Automated testing is the process through which automated tools run tests that repeat predefined actions, comparing a developing program's expected and actual outcomes. **More info:** 

Manual Testing	Automated Testing
Time consuming and tedious: Since test cases are executed by human resources so it is very slow and tedious.	Fast: Automation runs test cases significantly faster than human resources.
Less reliable: Manual testing is less reliable as tests may not be performed with precision each time because of human errors.	More reliable: Automation tests perform precisely same operation each time they are run.
Self-contained: Manual testing can be performed and completed manually and provide self-contained results.	Not self-contained: Automation can't be done without manual testing. You have to manually check the automated test results.
Implicit: Implicit knowledge are used to judge whether or not something is working as expected. This enables engineer to find extra bugs that automated tests would never find.	Explicit: Automated tests execute consistently as they don't get tired and/or lazy like us humans.

Verification and Validation

Are we building
the product **right**?

To ensure that work products meet their specified requirements.

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.



Positive and Negative

In **positive** testing
our intention is

to prove that an application will work on giving valid input data. i.e. testing a system by giving its corresponding valid inputs.



In **negative** testing
our intention is

to prove that an application will not work on giving invalid inputs.



Black-box, White-box, Grey-box

Black-box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester.

White-box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

Black-box Testing	White-box Testing
Levels Applicable To: System, Acceptance Test Levels	Levels Applicable To: Component, Integration Test Level
Responsibility: Quality Control Engineers	Responsibility: Software Developers



Grey-box Testing is a software testing method which is a combination of Black-box and White-box Testing methods.



Scripted and Unscripted

Scripted testing

Test execution carried out by following a previously documented sequence of tests.



Unscripted testing

Test execution carried out without previously documented sequence of tests.



Unscripted testing

Exploratory testing

An informal test design technique where the tester actively controls the design of the tests as those tests are performed and uses information gained while testing to design new and better tests

Ad-hoc testing

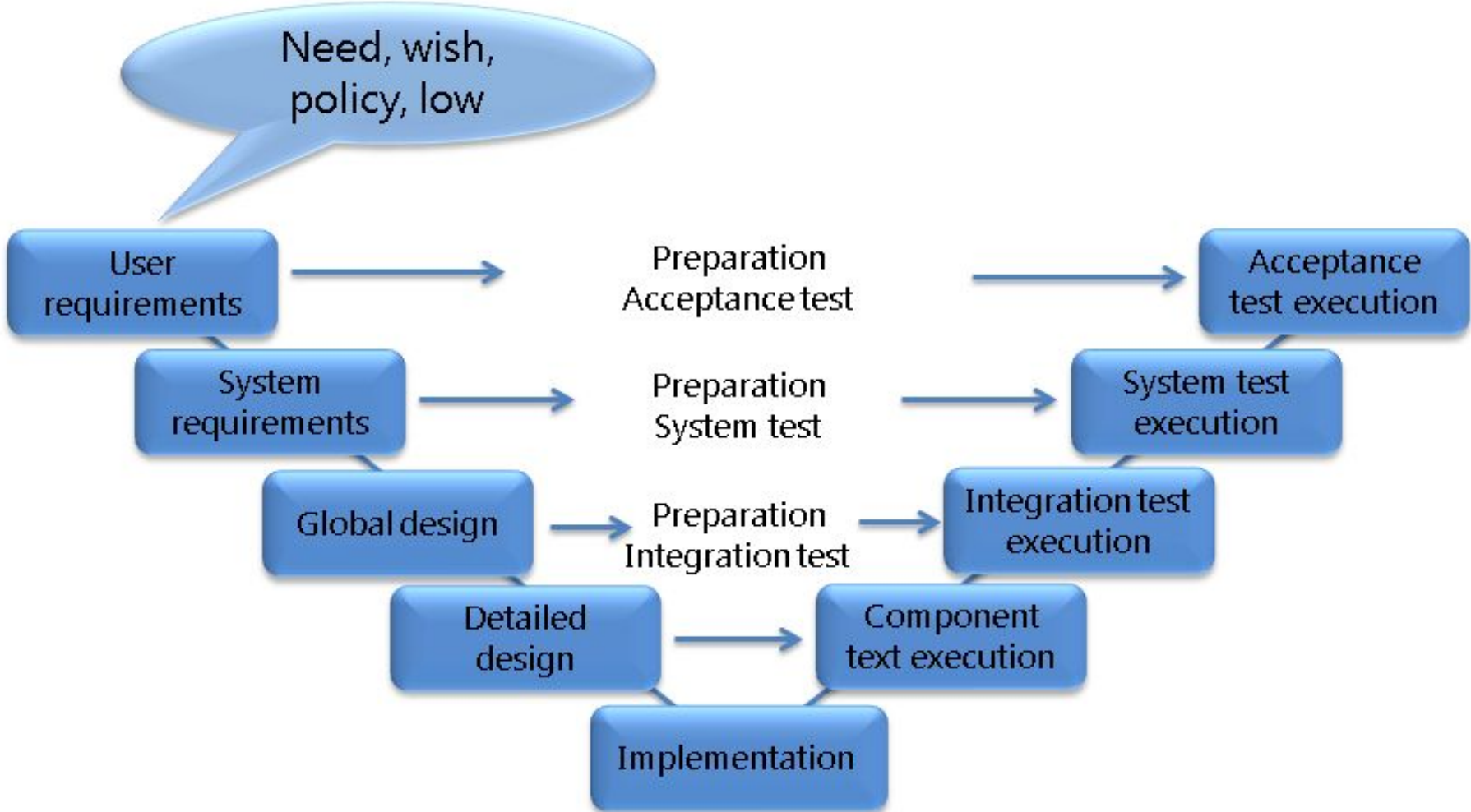
Testing carried out informally; no formal test preparation takes place, no recognized test design technique is used, there are no expectations for results and arbitrariness guides the test execution activity

Exploratory Testing	Ad-hoc Testing
Aim: to get the information to design new and better tests	Aim: to find defects
Result: defects are found and registered; new tests are designed and documented for further usage	Result: defects are found and registered



Test Types by Test Levels

Test Levels

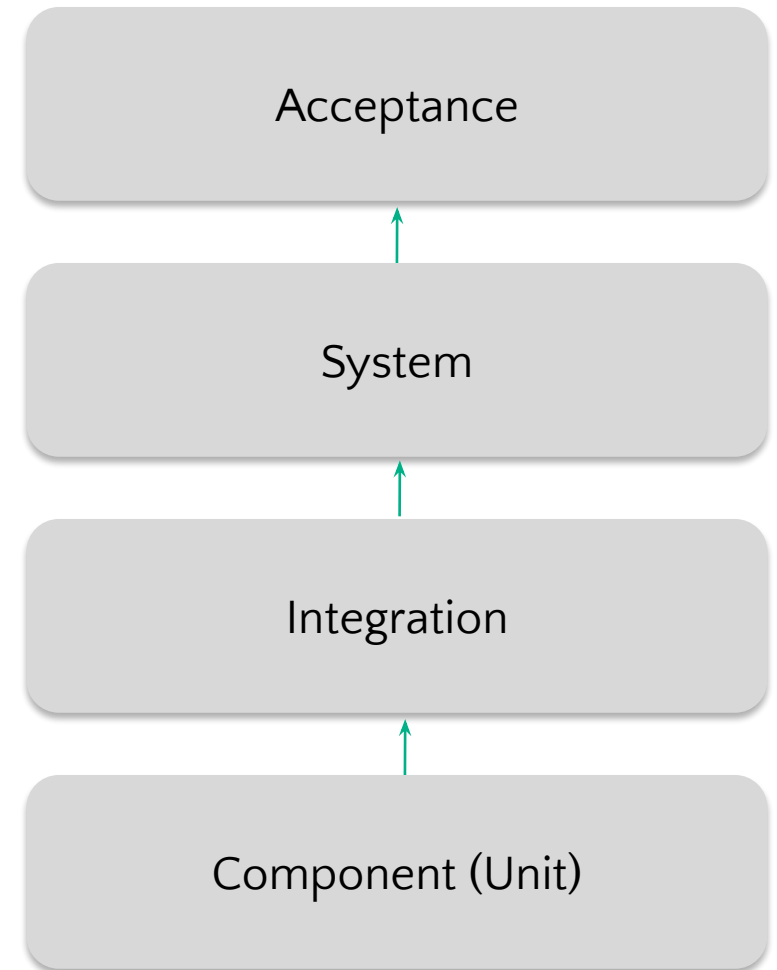


Test Levels

Test levels are groups of test activities that are organized and managed together.

According to ISTQB there are next test levels:

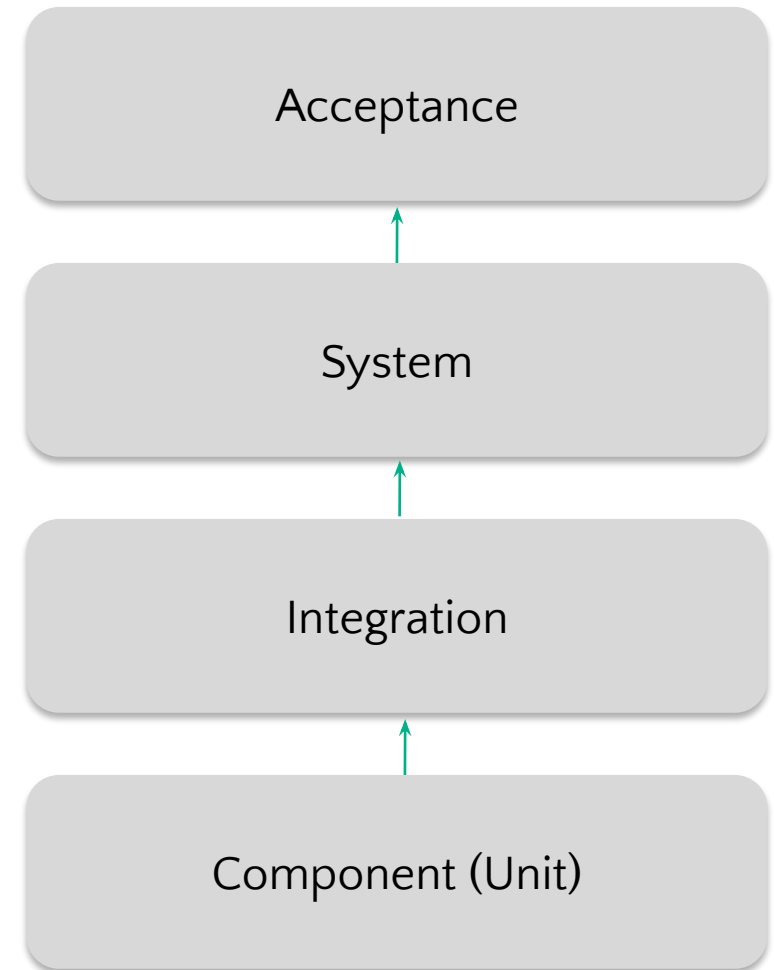
- Component testing
- Integration testing
- System testing
- Acceptance testing



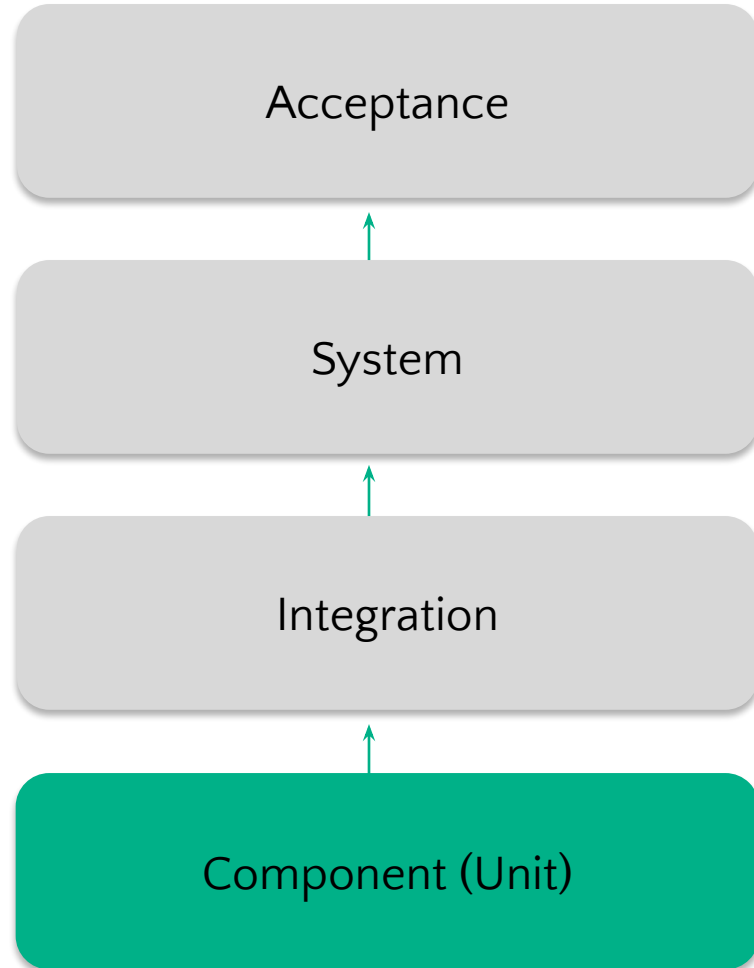
Test Levels

Test levels are characterized by the following attributes:

- Specific objectives
- Test basis, referenced to derive test cases
- Test object (i.e., what is being tested)
- Typical defects and failures
- Specific approaches and responsibilities



Component level



Testing on the Component Test Level is called **Component (Unit, Module)** testing

Component (Unit) Test Level	
Who	DEV
When	Component is developed
Why	To validate that each unit of the software performs as designed
How	White-box testing



softserve

Unit testing

Examples of a **test basis**:

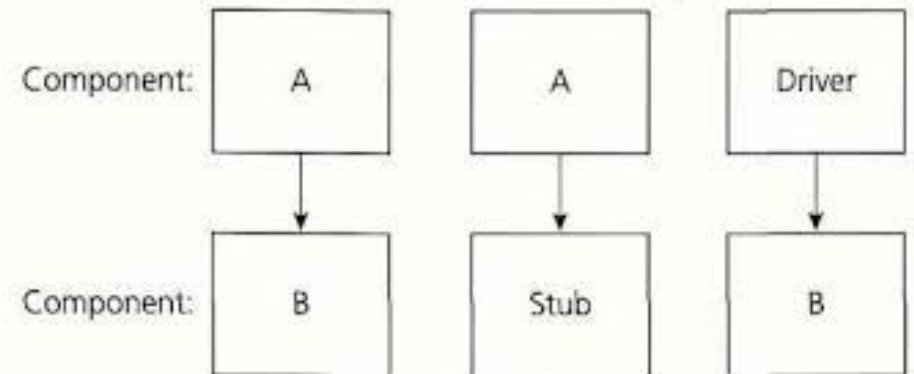
- Detailed design
- Code
- Data model
- Component specifications

Typical **test objects** for component testing include:

- Components, units or modules
- Code and data structures
- Classes
- Database modules

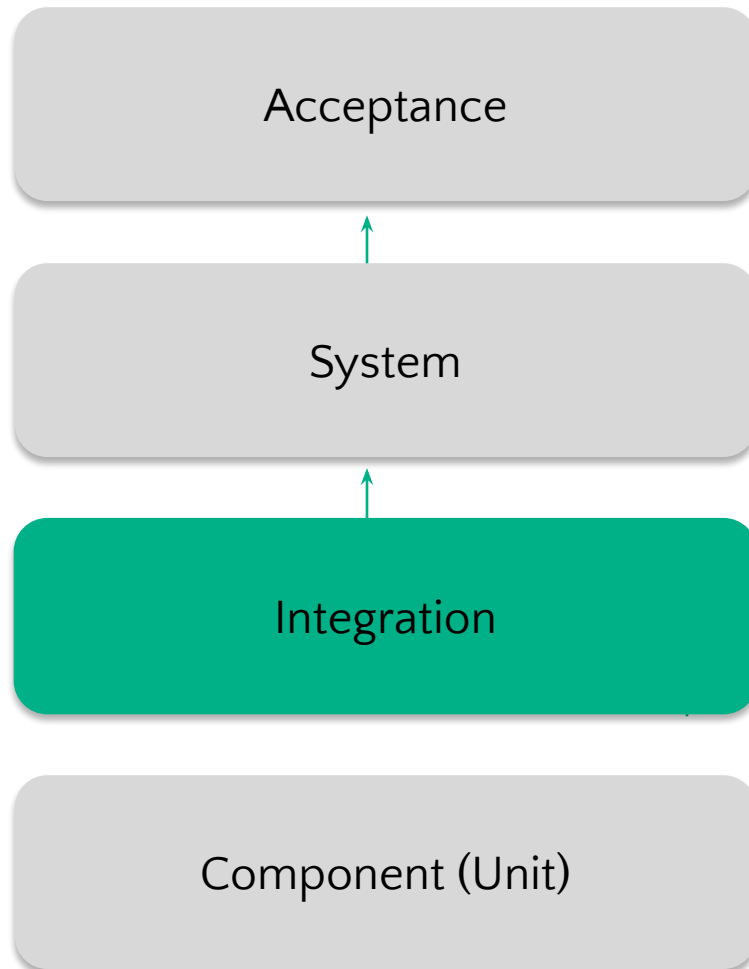
Typical **defects and failures**:

- Incorrect functionality (e.g., not as described in design specifications)
- Data flow problems
- Incorrect code and logic



softserve

Integration level



Testing on the Integration Test Level is called **Integration** testing

Integration Test Level	
Who	DEV, QC
When	Units to be integrated are developed
Why	To expose faults in the interaction between integrated units
How	White-box/ Black-box/ Grey-box Depends on definite units



Integration testing

Test objects:

- Subsystems
- Databases
- Infrastructure
- Interfaces
- APIs
- Microservices

Examples of a test basis:

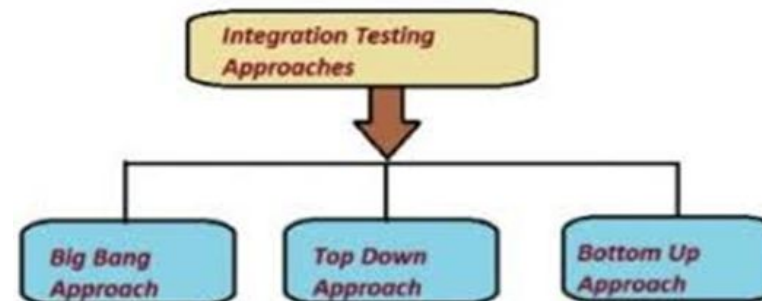
- Software and system design
- Sequence diagrams
- Use cases
- Architecture at component or system level
- Workflows

Typical defects for component integration testing

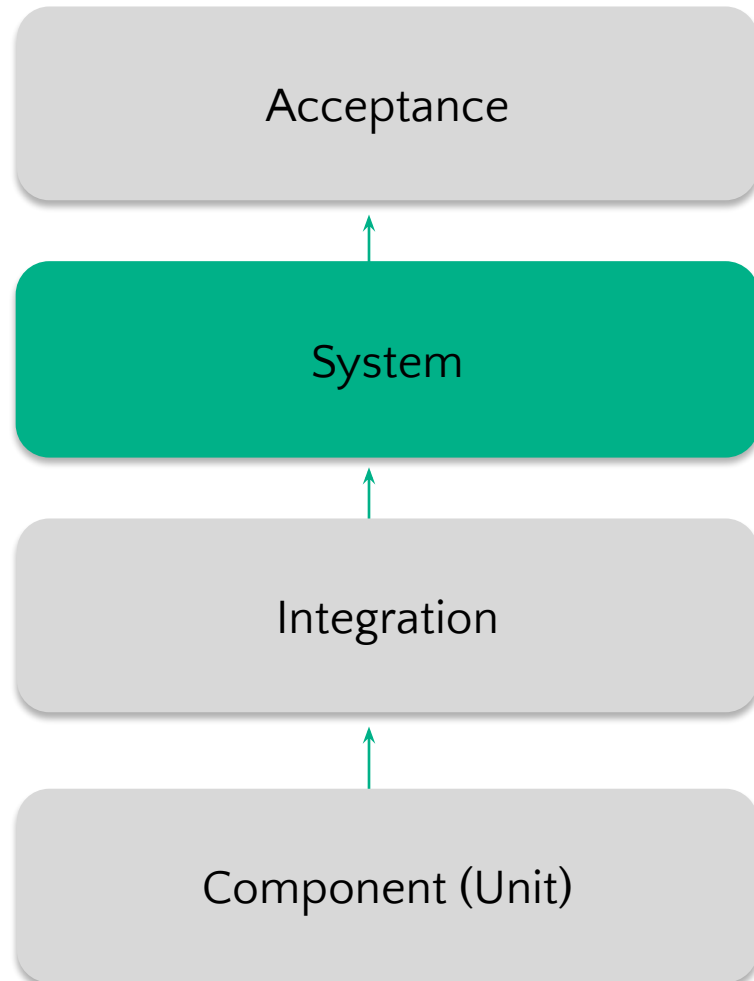
- Incorrect data, missing data, or incorrect data encoding
- Incorrect sequencing or timing of interface calls
- Interface mismatch
- Failures in communication between components

Typical defects for system integration testing

- Inconsistent message structures between systems
- Incorrect data, missing data, or incorrect data encoding
- Interface mismatch
- Failures in communication between systems
- Unhandled or improperly handled communication failures between systems



System level



Testing on the System Test Level is called **System** testing

System Test Level	
Who	QC
When	Separate units are integrated into System
Why	To evaluate the system's compliance with the specified requirements
How	Black-box testing



softserve

System testing

Examples of a **test basis**:

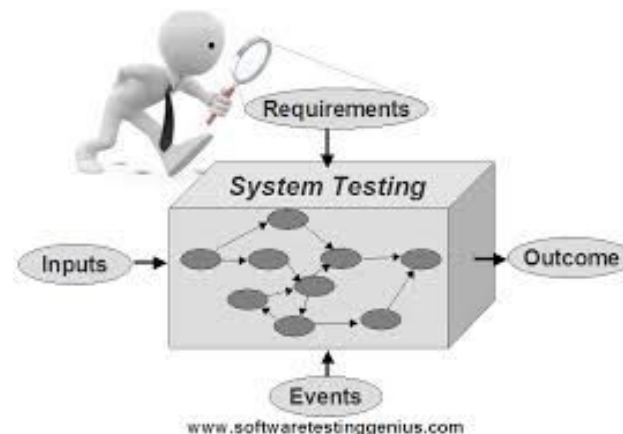
- System and SRS
- Risk analysis reports
- Use cases
- Epics and user stories
- State diagrams

Test objects:

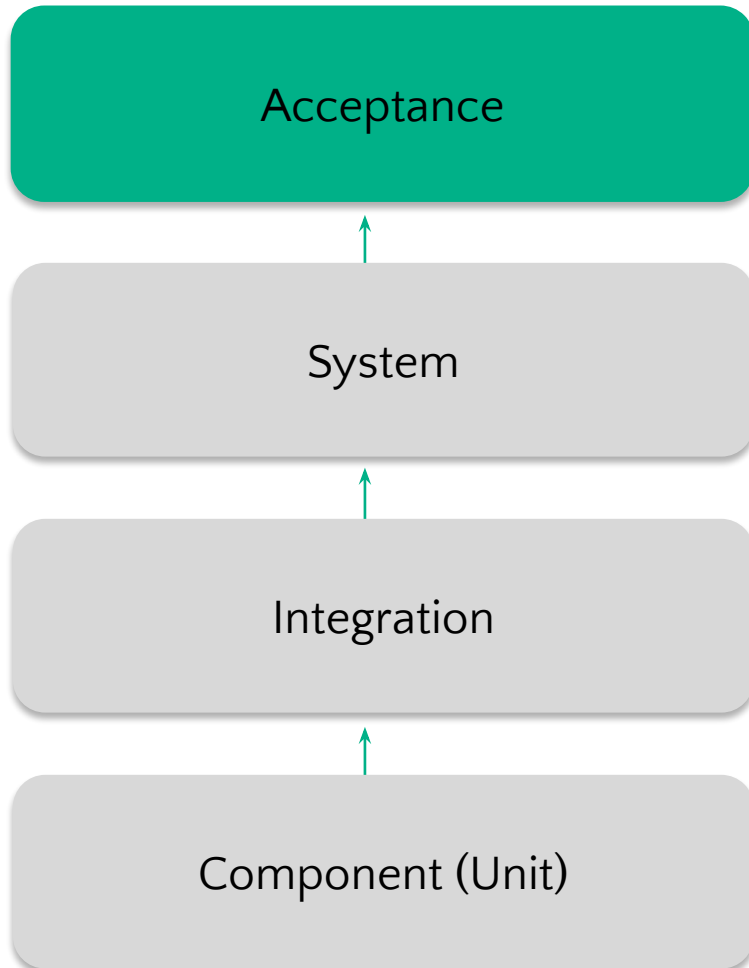
- Applications
- Hardware/software systems
- Operating systems
- System under test (SUT)
- System configuration and configuration data

Typical defects

- Incorrect calculations
- Incorrect or unexpected system functional or non-functional behavior
- Incorrect control and/or data flows within the system
- Failure of the system to work properly in the production environment(s)
- Failure of the system to work as described in system and user manuals



Acceptance level



Testing on the Acceptance Test Level is called **Acceptance** testing

Acceptance Test Level	
Who	People who have not been involved into development
When	Component is developed
Why	To evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery
How	Black-box testing



Acceptance testing

Examples of a **test basis**:

- Business processes
- User or business requirements
- Regulations, legal contracts and standards
- Use cases
- Installation procedures
- Risk analysis reports

Test objects:

- System under test
- System configuration and configuration data
- Business processes for a fully integrated system
- Recovery systems and hot sites
- Operational and maintenance processes
- Forms
- Reports
- Existing and converted production data

Typical defects

- System workflows do not meet business or user requirements
- Business rules are not implemented correctly
- System does not satisfy contractual or regulatory requirements
- Non-functional failures such as security vulnerabilities, inadequate performance efficiency under high loads, or improper operation on a supported platform

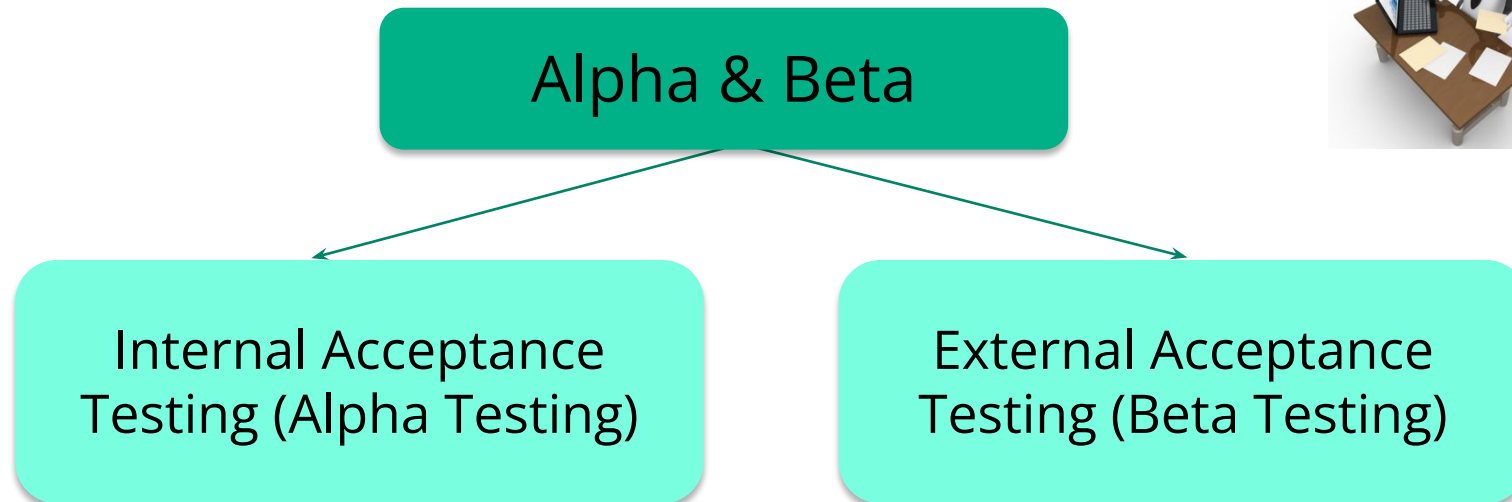


softserve

Acceptance Testing

Common **forms of acceptance testing** include the following:

- User acceptance testing
- Operational acceptance testing
- Contractual and regulatory acceptance testing
- Alpha and beta testing



Test Types

by Test Objectives

Test Types

How well?

Depending on its objectives, testing will be organized differently:

- Testing of a function to be performed by the component or system;
- Testing of a nonfunctional quality characteristic, such as reliability or usability;
- Testing of the structure or architecture of the component or system;
- Testing related to changes.

Are there any unintended changes?

What it does?



softserve

Test Types: Functional testing

Testing of function
(Functional testing)

Testing based on an analysis of the specification of the functionality of a component or system.

Testing of software product characteristics
(Non-functional testing)

Testing of software structure/architecture
(Structural testing)

Testing related to changes
(Confirmation and Regression testing)

According to [ISO 25010](#) **Functional suitability** consists of:

- Functional completeness
- Functional correctness
- Functional appropriateness

Functional testing. Attributes.

Functional suitability: the degree to which a component or system provides functions that meet stated and implied needs when used under specified conditions

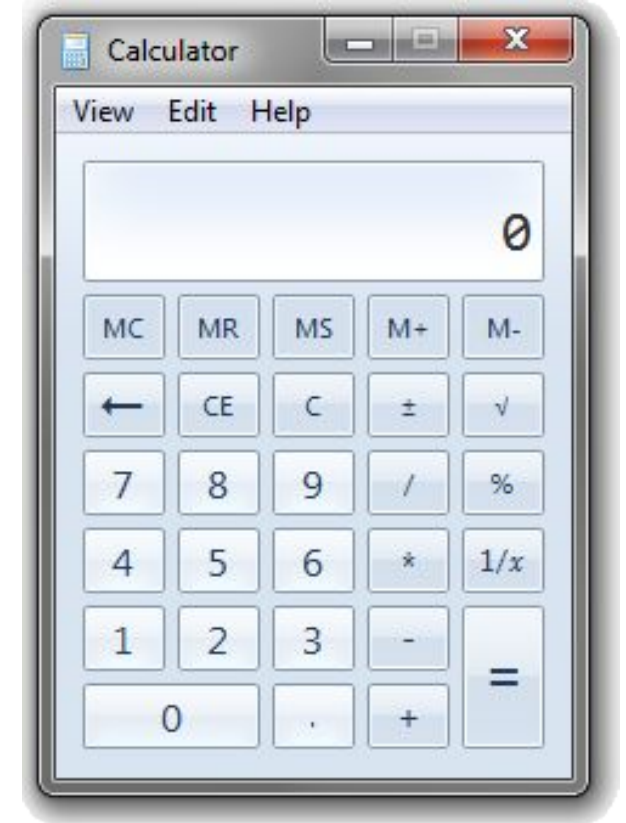
- **Functional completeness:** degree to which the set of functions covers all the specified tasks and user objectives
- **Functional correctness:** degree to which a product or system provides the correct results with the needed degree of precision
- **Functional appropriateness:** degree to which the functions facilitate the accomplishment of specified tasks and objectives



oftserve

Functional testing Example #1

- Verify adding of two numbers ($5+3$ should be 8);
- Verify subtraction of two numbers ($5-2$ should be 3);
- Verify multiplication of two number ($5*3$ should be 15);
- Verify division of two numbers ($10/2$ should be 5);
- Verify getting radical of some number ($\sqrt{25}$ should be 5);
- Verify multiplication of some number by zero ($5*0$ should be 0);
- Etc.



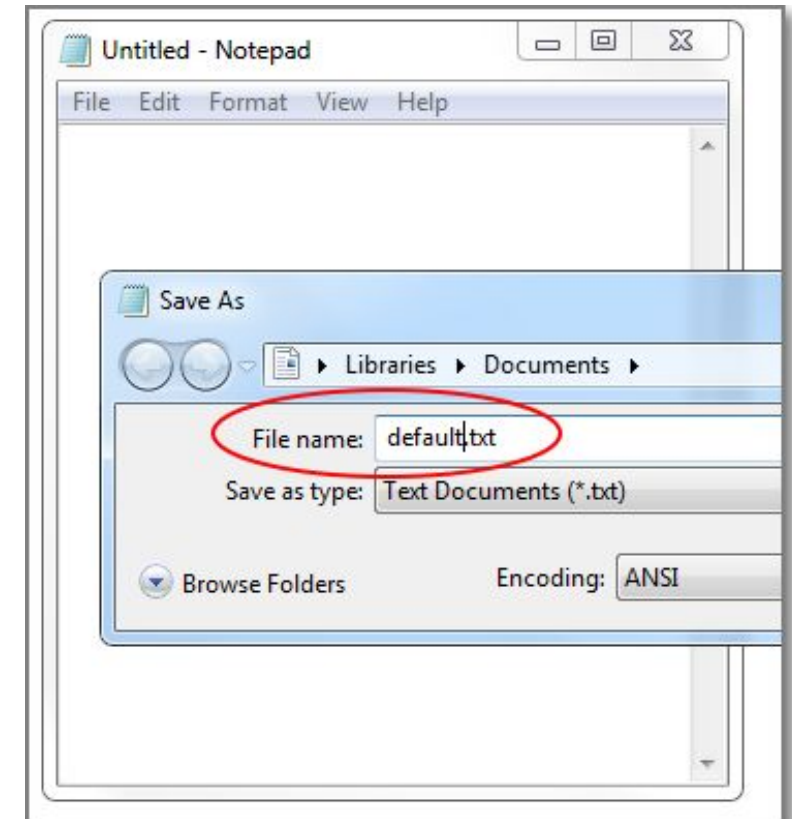
softserve

Functional testing Example #2

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.

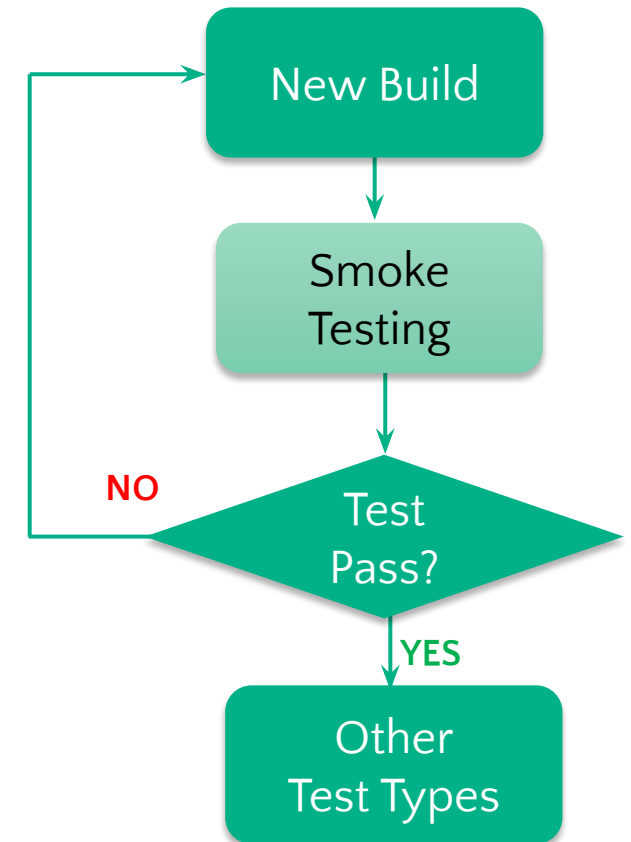


Functional testing: Smoke

A subset of all defined/planned test cases that cover the main functionality of a component or system, to ascertaining that the most crucial functions of a program work, but not bothering with finer details.

Purposes:

- ✓ is done before accepting a build for further testing;
- ✓ is intended to reveal simple but critical failures to reject a software build\release;
- ✓ determines whether the application is so badly broken that further testing is unnecessary.



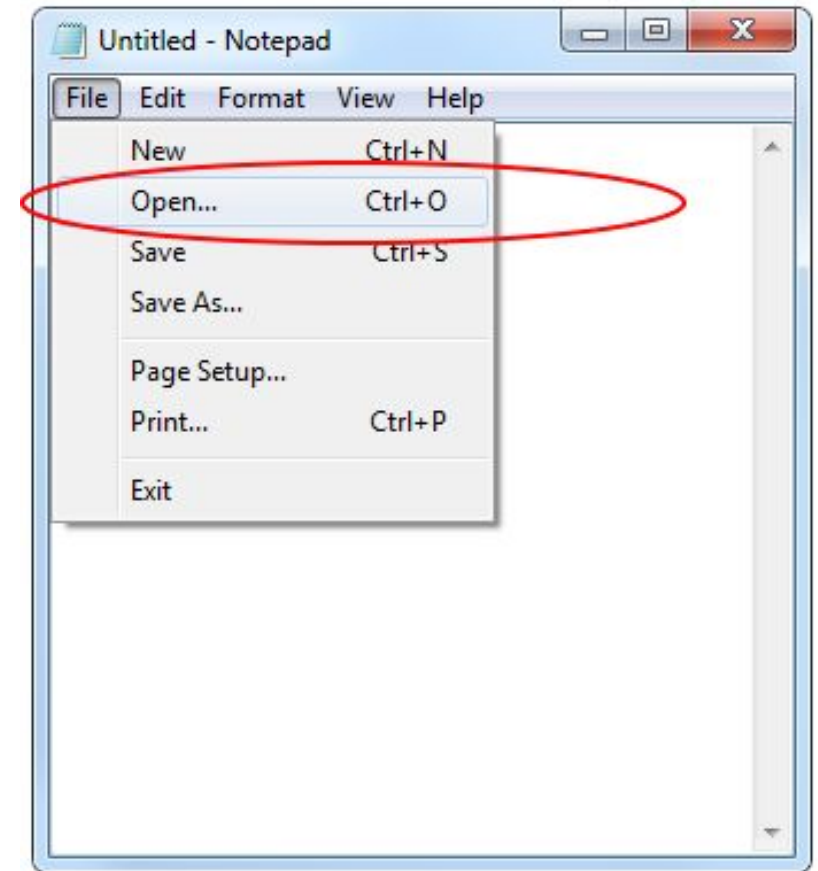
Smoke testing Example

Task: Test new version of a Notepad application.

Smoke Testing Procedure: quickly check the main Notepad features (run application, type text, open file, edit file, save file).

Defect: There is no ability to Open a file. Button "Open" does nothing.

Summary: build is not accepted, critical bug is logged to a Bug Tracking system, developers team and project manager are informed by QC engineer about that fact.



Test Types: Non-functional testing

Testing of function
(Functional testing)

Testing of software product
characteristics
(Non-functional testing)

Testing of software
structure/architecture
(Structural testing)

Testing related to changes
(Confirmation and Regression
testing)

Testing the attributes of a component or system that do not relate to functionality.

According to [ISO 25010](#) **Non-functional characteristics** are:

- Performance efficiency
- Compatibility
- Usability
- Reliability
- Security
- Maintainability
- Portability

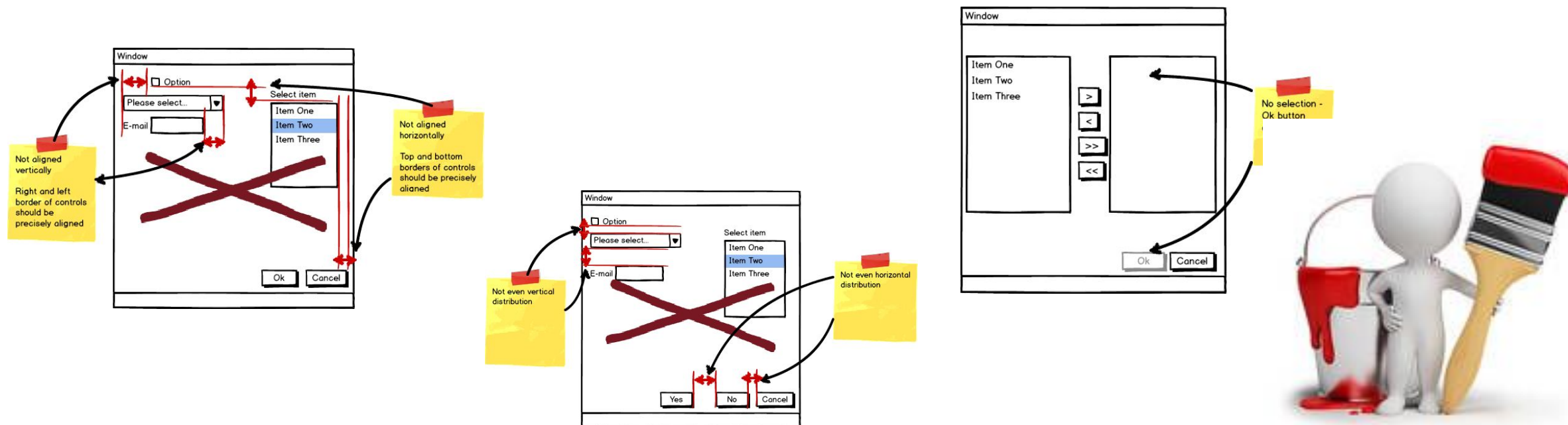
Non-functional testing

- **Performance efficiency:** Time behavior, Resource utilization, Capacity.
- **Compatibility:** Co-existence, Interoperability.
- **Usability:** Appropriateness recognizability, Learnability, Operability, User error protection, User interface aesthetics, Accessibility.
- **Reliability:** maturity (robustness), fault-tolerance, recoverability and availability.
- **Security:** Confidentiality, Integrity, Non-repudiation, Accountability, Authenticity.
- **Maintainability:** Modularity, Reusability, Analysability, Modifiability and Testability.
- **Portability:** Adaptability, Installability and Replaceability.



Non-functional testing: UI

UI Testing: The testing a product's graphical user interface to ensure it meets its written specifications



Check if any UI recommendations exist for the application type your team develop. Make sure dialogs you test comply with these recommendations.

Non-functional testing: Performance

Performance Testing: Testing with the intent of determining how efficiently a product handles a variety of events.



Purposes:

- ✓ demonstrate that the system meets performance criteria;
- ✓ compare two systems to find which performs better;
- ✓ measure what parts of the system or workload cause the system to perform badly.



softserve

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: check response time of the server with 100 and 300 users at the same time.

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



softserve

Non-functional testing: Load

Load testing is a type of performance testing conducted to evaluate the behavior of a component or system with increasing load, e.g. numbers of parallel users and/or numbers of transactions, to determine what load can be handled by the component or system.

Purposes

- ✓ evaluation of performance and efficiency of software
- ✓ performance optimization (code optimization, server configuration)
- ✓ selection of appropriate hardware and software platforms for the application



softserve

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. Verify that the server is able to respond correctly with the maximum number of users.

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



softserve

Non-functional testing: Stress

Stress testing: 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

Purposes:

- ✓ the general study of the behavior of the system under extreme loads
- ✓ examination of handling of errors and exceptions under extreme load
- ✓ examination of certain areas of the system or its components under the disproportionate load
- ✓ testing the system capacity



softserve

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 501, 510, and 550 concurrent users. Verify that `500+` users get appropriate error message from server. (e.g. *`an error occurred, please try again later`*) The server should not crash due to overload.

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.



softserve

Non-functional testing: L10N, I18N

Localization is the process of adapting a globalized application to a particular culture/locale.



Localization (L10N) testing checks how well the application under test has been Localized into a particular target language.



Internationalization is the process of designing and coding a product so it can perform properly when it is modified for use in different languages and locales.



Internationalization (I18N) testing checks if all data/time/number/currency formats are displayed according to selected locale and if all language specific characters are displayed.



Localization testing Example



Task: Verify that 'Login' page is translated to German

Localization Testing Procedure:

Test all labels and captions on the page whether they are translated to German; force appearance of different messages (e.g.: when password or login does not exist) to check whether they are localized and not truncated

Defects:

- "Password you have entered does not exist in the system" message is truncated on German Locale;
- "Login" label is not translated and still appears in English under German locale.

Internationalization testing Example

Task:

Verify that list of users with German special characters (e.g.: "ü", "ß" etc) in names are sorted correctly by 'First Name' column

Functional Testing Procedure:

Create enough different users with special characters in First Name. Sort them via the table to ensure that special characters are sorted correctly

Defect:

Sorting performs incorrectly: all names which start from special characters (e.g.: "ü", "ß" etc) are always listed at the end of the sorted column. Instead they should be sorted with all other characters (e.g.: "ß" at once after word with "ss", but on at the end of the list)



Test Types: Structural testing

Testing of function
(Functional testing)

Testing of software product
characteristics
(Non-functional testing)

Testing of software
structure/architecture
(Structural testing)

Testing related to changes
(Confirmation and Regression
testing)



Mostly applied at Component
and Integration Test Levels

softserve

Test Types: Confirmation and Regression

Testing of function
(Functional testing)

Testing of software product
characteristics
(Non-functional testing)

Testing of software
structure/architecture
(Structural testing)

Testing related to changes
(Confirmation and
Regression testing)

If we have made a change to the software, we will have changed the way it functions, the way it performs (or both) and its structure.



Test Types: Confirmation

Confirmation testing or re-testing is a testing type that runs test cases that failed the last time they were run, in order to verify the success of corrective actions.

1. Build 1.0.0 – Test for Function A - Passed, test for Function B - Failed
2. In the next build 1.0.1 changes are introduced to Function B and Common Library by developers
3. Now we need to re-run test for Function B to ensure, that Function B was changed correctly.



Test Types: Regression

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. It is performed when the software or its environment is changed.

Purpose:

- ✓ verifies that the system still meets its requirements

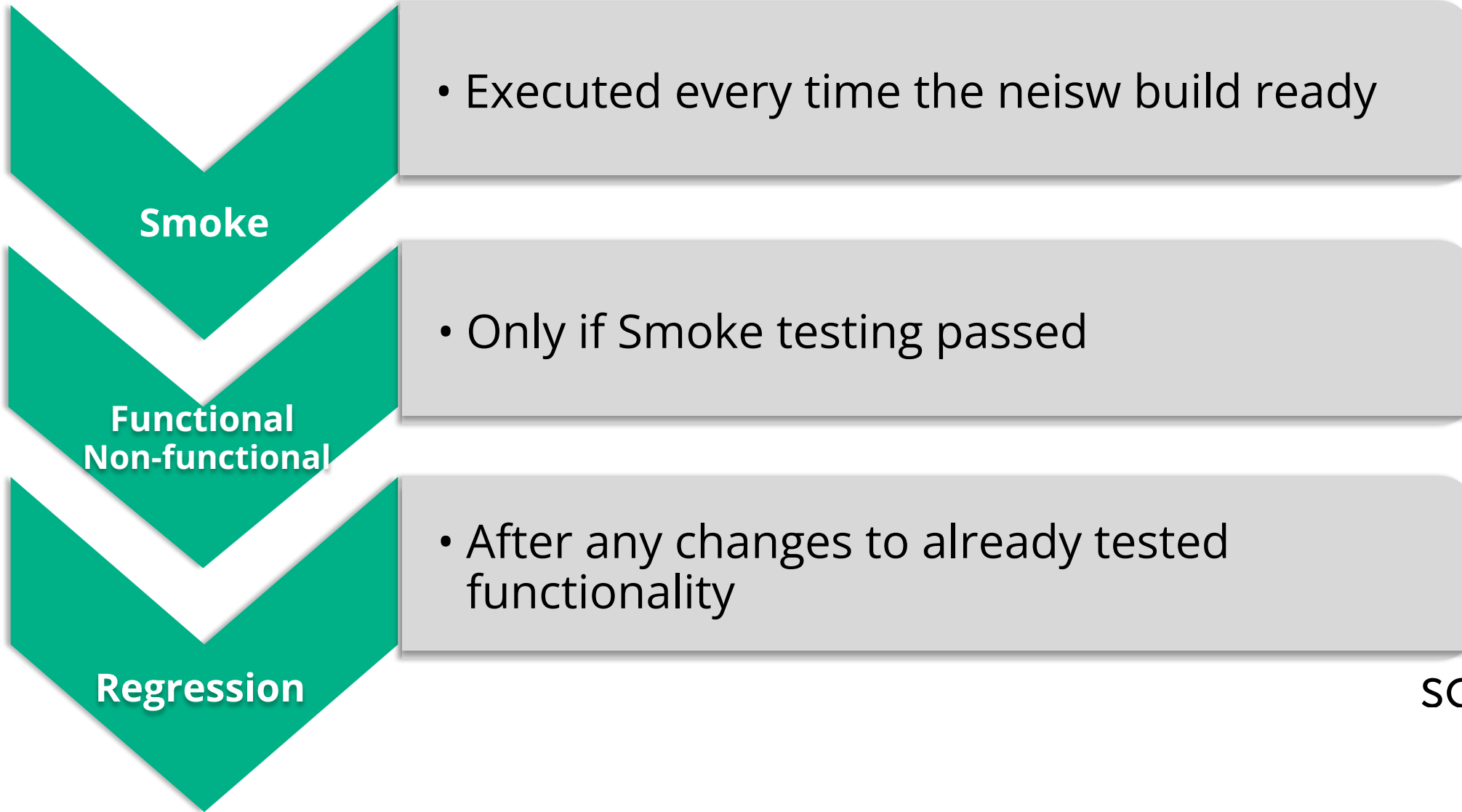


May be any type of software testing (functional, GUI, etc...)

softserve

Testing Order

Testing Order



Testing Order

Some factors to consider in prioritizing test cases:

- ✓ Mission-critical components
- ✓ Complex features
- ✓ Where failures would be most visible
- ✓ Features that undergo frequent changes
- ✓ Areas with past histories of problems
- ✓ Areas with complex coding
- ✓ Areas of most frequent use



Summary

Test activities can be grouped using different classification:

- By the degree of automation (Manual and Automated);
- By the level of awareness about the system and its internal structure (Black-, White-, Grey-box);
- By the basis of positive scenario (Positive and Negative);
- By the degree of preparedness to be tested (Scripted and Unscripted);
- By the degree of component isolation (by Test levels);
- By the Test Objectives.

All mentioned Test Types are not mutually exclusive, but are complementary.

Revision History

Version	Date	Remark	Author
v.1	June, 2017		M. Harasym
v.2	October, 2018	Update according to new ISTQB Standard	V. Ryazhska

Thank you

softserve