2 : Software Testing Process

IT6206 – Software Quality Assurance

Level III - Semester 6

Overview

- Effective and efficient testing requires test approaches that are properly planned and carried out, with tests designed and implemented to cover proper areas of the system, executed in the right sequence and with their results reviewed regularly.
- This is a process, with tasks and activities that can be identified and need to be done.

Intended Learning Outcomes

At the end of this lesson, you will be able to;

- Describe and explain definitions and activities related to software testing
- Explain the relationships between software development activities and test activities in the software development life cycle.
- Compare different test levels based on objectives, test basis, test objects, typical defects and failures, and approaches and responsibilities.
- Compare functional and non functional testing
- Compare the purpose of confirmation testing and regression testing.
- Compare static and dynamic testing

List of sub topics

2.1 Testing Process

Image Source : https://www.indiumsoftware.com/blog/why-software-testing/

2.1 Testing Process

- Testing is not just test execution
- Effective and Efficient testing requires test approaches that properly planned and carried out,
    - with tests designed and implemented to cover the proper areas of the system
    - executed in the right sequence
    - with their results reviewed regularly
- This is a “Process”, with tasks and activities that can be identified and need to be done.

Continued to next slide

2.1 Testing Process ctd.

- There is no “one size fits all” test process, but testing does need to include common set of activities, or it may not achieve its objectives.
- The “right” test process for you is one that achieves your test objectives in the most efficient way
- The best process for you would not be best for another (and vice versa)

2.1 Testing process ctd.

The factors that influence a particular test process include the following (not a fully comprehensive list):

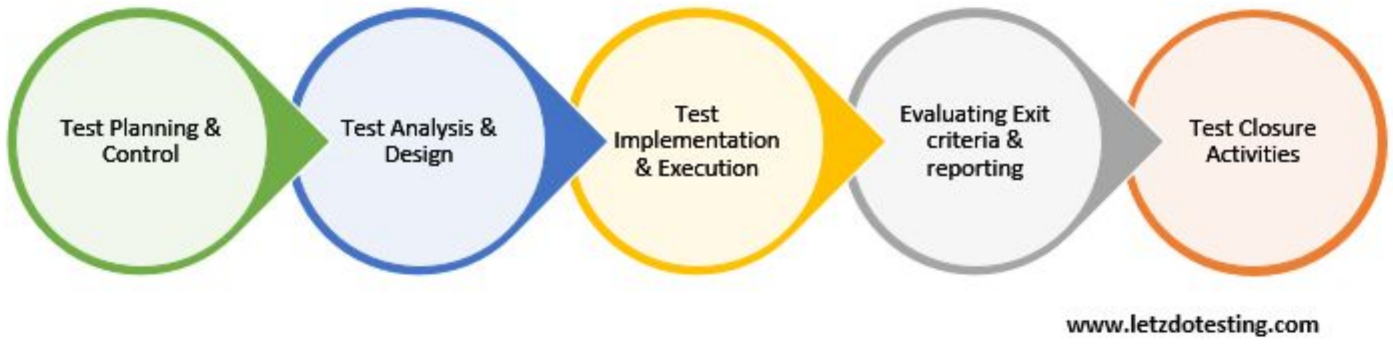
- Software development life cycle model and project methodologies being used.
  - Ex – Test process of an agile project developing mobile apps would be quite different from test process of an organization producing medical devices.
- Test levels and test types being considered
  - Ex – A large complex project may have several types of integration testing, with a test process reflecting that complexity.

List continued to next slide

2.1 Testing process ctd.

- Product and project risks (lower the risk, the less formal the process needs to be and vice versa).
- Business domain
- Operational constraints (budgets, resources, timescales, complexity and etc.)
- Organizational policies and practices
- Required internal and external standards

2.2 Test Activities and Tasks



2.2 Test Activities and Tasks

A test process consists of the following main groups of activities:

- Test planning
- Test monitoring and control
- Test analysis
- Test design
- Test implementation
- Test execution
- Test completion

- These activities appear to be logically sequential, in the sense that tasks within each activity often create the preconditions or precursor work products for tasks in subsequent activities.
- However, The activities in the process may **overlap** or take place **concurrently** or **iteratively**.

2.2 Test Activities and Tasks ctd.

Test Planning

- **Test planning** involves defining the objectives of testing and the approach for meeting those objectives within the project constraints and contexts.
- This includes:
  - Deciding on suitable test techniques to use.
  - Deciding what tasks need to be done.
  - Formulating a test schedule and other things.

2.2 Test Activities and Tasks ctd.

Test monitoring and control

Test Monitoring - A test management activity that involves,

- Checking the status testing activities,
- Identifying any variance from the planned or expected status,
- And, reporting status to stakeholders

- If things are going very wrong it may be time to stop the testing or even stop the project completely.
- Therefore one way we can monitor test progress is by using “exit criteria”, also known as “definition of done” in Agile development

2.2 Test Activities and Tasks ctd.

The **exit criteria** for test execution might include:

- Checking test results and logs against specified coverage criteria (we have not finished testing until we have tested what we planned to test).
- Assessing the level of component or system quality based on test results and logs (e.g. the number of defects found or ease of use).
- Assessing product risk and determining of more tests are needed to reduce the risk on acceptable level.

Test Control – A test management task that deals with developing and applying a set of corrective actions to get a test project on track when monitoring shows a deviation from what was planned.

2.2 Test Activities and Tasks ctd.

<u>Test Analysis</u>

- The activity that identifies test conditions by analyzing the **test basis.**
- The test basis is analyzed to identify testable features and define associated **test conditions.**
- In test analysis, more general testing objectives defined in the test plan is transformed to tangible test conditions.

Key words :

Test condition – An aspect of test basis that is relevant in order to achieve specific test objectives.

2.2 Test Activities and Tasks ctd.

Test Basis - The body of knowledge used as the basis for test analysis and design.

In simple terms the test basis is:

Everything upon which we base our tests.

Examples of a test basis include:

- Requirement specifications
- Design and implementation information
- The implementation of the component or system it self
- Risk analysis report

2.2 Test Activities and Tasks ctd.

- Test analysis includes the following major activities and tasks:
    - Analyze the test basis appropriate to the test level being considered.
    - Evaluate the test basis and test items to identify various types of defects that occur such as:
        - Ambiguities
        - Omissions
        - Inaccuracies
        - Inconsistencies
        - Contradictions
        - Superfluous statements

Activities continued to next slide

2.2 Test Activities and Tasks ctd.

- Identify features and sets of features to be tested.
- Identify and prioritize test conditions for each feature.
- Identify and prioritize test conditions for each feature.
- Capture bi-directional traceability between each element of the test basis and the associated test conditions.

2.2 Test Activities and Tasks ctd.

- Test techniques such as black box testing, white box testing and experience – based techniques could be used to identify test conditions.
- One of the most beneficial side effects of identifying what to test in test analysis is that you will find defects.
  - E.g. Missing requirements, Contradictory statements between different documents and etc.
- Test analysis can also help to validate whether the requirements properly capture customer, user and other stakeholder needs.

2.2 Test Activities and Tasks ctd.

<u>Test Design</u>

- The activity of deriving and specifying test cases from test conditions.
- Addresses the question “how to test”; that is,
    - what specific inputs and data are needed in order to exercise the software for a particular test condition.
- In test design, test conditions are elaborated(at a high level) in test cases, set of test cases and other testware
- Test analysis identifies general ‘things’ to test, and test design makes these general things specific for the component or system that we are testing.

Just as in test analysis, test design can also identify defects in the test basis and in the existing test conditions.

2.2 Test Activities and Tasks ctd.

Test design includes the following major activities

- Design and prioritize test cases and sets of test cases.
- Identify the necessary **test data** to support the conditions and test cases as they are identified and designed.
- Design the test environment, including set-up, and identify any required infrastructure and tools.
- Capture bi-directional traceability between the test basis, test conditions, test cases and test procedures.

2.2 Test Activities and Tasks ctd.

Test Implementation

- The activity that prepares the testware needed for test execution based on test analysis and design.
- **Test procedures** are specified in test implementation.
- Test implementation also involves setting up the test environment and anything else that need to be done to prepare for test execution, such as creating testware.
- Test implementation asks "do we now have everything in place to run the tests?"

Keywords:

Test Procedure – A sequence of test cases in execution order, and associated actions that may be required to set up the initial preconditions and wrap-up activities post execution.

2.2 Test Activities and Tasks ctd.

Test implementation includes the following major activities:

- Develop and prioritize the test procedures and, potentially create automated test scripts.
- Create **test suites** from the test procedures and automated test scripts(if any).
- Arrange the test suites within a **test execution schedule** in a way that results in efficient test execution.
- Build the test environment (possibly including test harnesses, service virtualization, simulators and other infrastructure items) and verify that everything needed has been set up correctly.
- Prepare test data and ensure that it is properly loaded in the test environment (including inputs, data resident in database and other data repositories, and system configuration data).
- Verify and update the bi-directional traceability between the test basis, test conditions, test cases, test procedures and test suites.

2.2 Test Activities and Tasks ctd.

Keywords:

Test suites – A set of test cases or test procedures to be executed in a specific test cycle.

Test execution schedule – A schedule for thee execution of test suites within a test cycle

2.2 Test Activities and Tasks ctd.

Test execution

- The process of running a test on the component or system under test, producing actual results.
- In test execution, the test suites that have been assembled in test implementation are run, according to the test execution schedule.

Keywords:

Testware – Work products produced during the test process for use in planning, designing, executing, evaluating and reporting on testing.

2.2 Test Activities and Tasks ctd.

Test execution includes the following major activities:

- Record the identities and versions of all of the items, test objects, test tools and other **testware.**
- Execute the tests either manually or by using a automated test execution tool, according to the planned sequence.
- Compare actual results with expected results, observing where the actual and expected results differ. Those are caused by anomalies.
- Analyze the anomalies in order to establish their likely causes. Failures may occur due to defects in the code or they may be false-positives. A failure may also be due to test defect, such as defects in specified test data, in a test document or test environment, or simply due to mistake in the way the test was executed.

2.2 Test Activities and Tasks ctd.

- Report defects based on the failures observed
- Log the outcome of test execution(e.g. pass, fail or blocked). This includes not only the anomalies observed and the pass/fail status of the test cases, but also the identities and versions of the software under test, test tools and testware.
- As necessary, repeat the test activities when actions are taken to resolve the discrepancies.
- Verify and update the bi-directional traceability between the test basis, test conditions, test cases, test procedures and test results.

2.2 Test Activities and Tasks ctd.

Test completion

- The activity that makes test assets available for later use, leaves test environment in a satisfactory condition and communicates the results of testing to relevant stakeholder.
- Test completion activities collect data from completed test activities to consolidate experience, testware and any other relevant information.
- Test completion activities should occur at major milestones.
- The specific milestones that involve test completion activities should be specified in the test plan.

2.2 Test Activities and Tasks ctd.

Test completion includes the following major activities:

- Check whether all defect reports are closed, entering change requests or product backlog items for any defects that remain unresolved at the end of the test execution.
- Create a test summary report to be communicated to stakeholders.
- Finalize and archive the test environment, the test data, the test infrastructure and other testware for later reuse.

Activities continued to next slide

2.2 Test Activities and Tasks ctd.

- Hand over the testware to the maintenance teams, other project teams, and/or other stakeholders who could benefit from its use.
- Analyze lessons learned from completed test activities to determine changes needed for future iterations, releases and projects.
- Use the information gathered to improve the test process maturity, especially as an input to test planning for future projects.

2.2 Test Activities and Tasks ctd.

Test Work Products

- **Work products** is the generic name given to any form of documentation, informal communication or artifact that is used in testing.
- Test work products are created as part of the test process and there is significant variation in the types of work products created, in the way they are organized and managed and in the names used for them.

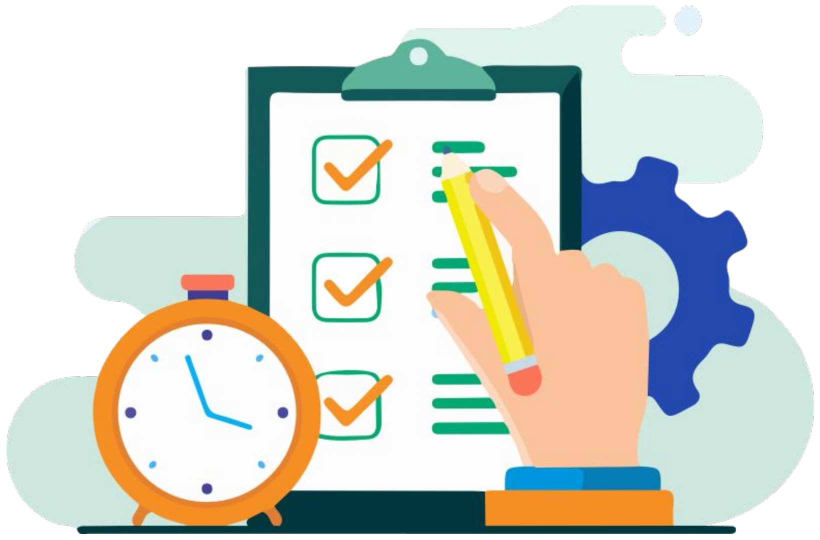
2.2 Test Activities and Tasks ctd.

- Types of work products
  - Test planning work products
    - Test plans
  - Test monitoring and control work products
    - Test reports
  - Test analysis work products
    - Test Conditions
  - Test design work products
    - Test cases and sets of test cases
  - Test implementation work products
    - Test procedures and the sequencing of those procedures
    - Test suites
    - A test execution schedule

2.2 Test Activities and Tasks ctd.

- Test execution work products
  - Documentation of the status of individual test cases or test procedures
  - Defect reports
  - Documentation about which item(s), test object(s), test tools and testware were involved in the testing.
- Test completion work products
  - Test summary reports
  - Action items for improvement of subsequent projects or iterations
  - Change requests or product backlog items
  - Finalized test ware

2.3 Traceability between the test basis and test work products



2.3 Traceability between the test basis and test work products

- Traceability – “The degree to which a relationship can be established between two or more work products”.
- There is **bi-directional** traceability for all test work products covered in previous slides.

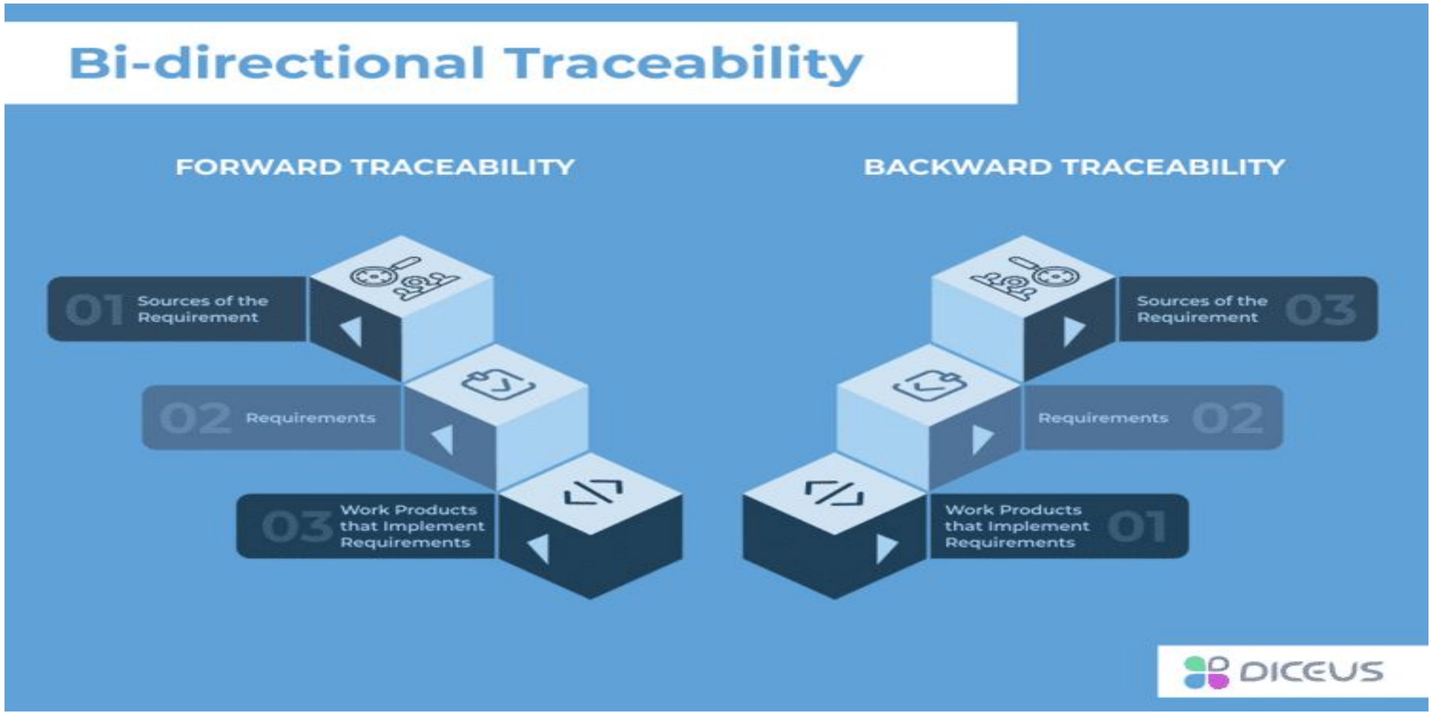
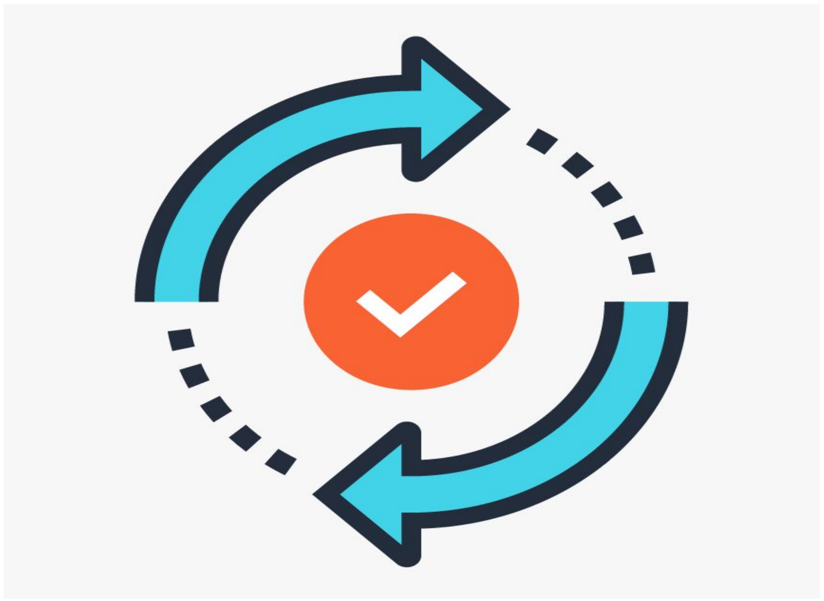

Bi-directional Traceability
FORWARD TRACEABILITY
01 Sources of the Requirement
02 Requirements
03 Work Products that Implement Requirements
BACKWARD TRACEABILITY
Sources of the Requirement 03
Requirements 02
Work Products that Implement Requirements 01
DICEUS

Image Source : https://diceus.com/what-is-a-traceability-matrix/

2.3 Traceability between the test basis and test work products ctd.

- Good traceability also supports the following
  - Analyzing the impact of changes
  - Making testing auditable, and being able to measure coverage.
  - Meeting IT governance criteria(where applicable)
  - Improving the coherence of test progress reports and test summary reports to stakeholders
  - Relating the technical aspects of testing to stakeholders in terms that they can understand
  - Providing information to assess product quality, process capability and project progress against business goals.

2.4 Software Development and Software Testing



Source: https://www.kindpng.com/imgv/oTTxJh_software-development-life-cycle-icon-hd-png-download//

2.4 Software Development and Software Testing

This section will discuss software development models and how testing fits into them.

- The life cycle model that is adopted for a project will have a big impact on the testing that carried out.
- The way testing is organized must fit the development life cycle or it will fail to deliver its benefits.
- For example, if time to market is the key driver, then the testing must be fast and efficient. If a fully documented software development life cycle, with an audit trail of evidence is required, the testing must be fully documented.

2.4 Software Development and Software Testing ctd.

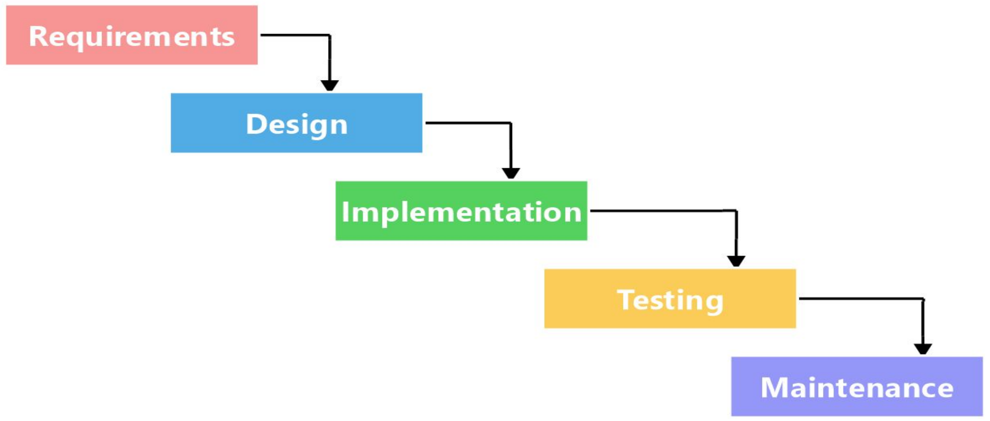
Which ever life cycle model is being used, there are several characteristics of good testing:

- For every development activity there is a corresponding test activity.
- Each test level has test objectives specific to that level.
- The analysis and design of tests for a given test level should begin during the corresponding software development activity.
- Testers should participate in discussions to help define and refine requirements and design. They should also be involved in reviewing work products as soon as drafts are available in the software development cycle.

2.4 Software Development and Software Testing ctd.

Sequential Development model

- Waterfall Model



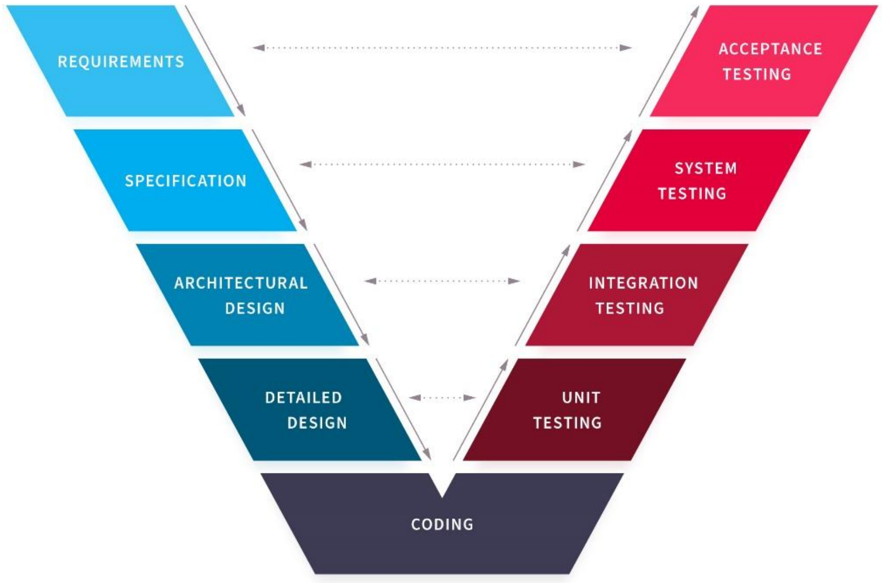
Source https://blog.mindmanager.com/waterfall-methodology/

Testing tends to happen at the end of the cycle, so defects are detected close to the live implementation date.

With this model it is difficult to get feedback passed backwards up the waterfall and there are difficulties if we need to carry out numerous iterations for a particular phase.

2.4 Software Development and Software Testing ctd.

- The V-model



Source: https://nix-united.com/blog/software-development-life-cycle-nix-approach-to-sdlc/

For every single phase in the development cycle, there is directly associated testing phase.

V-model provides guidance on how testing begins as early as possible in the life cycle and it also shows that testing is not only an execution-based activity.

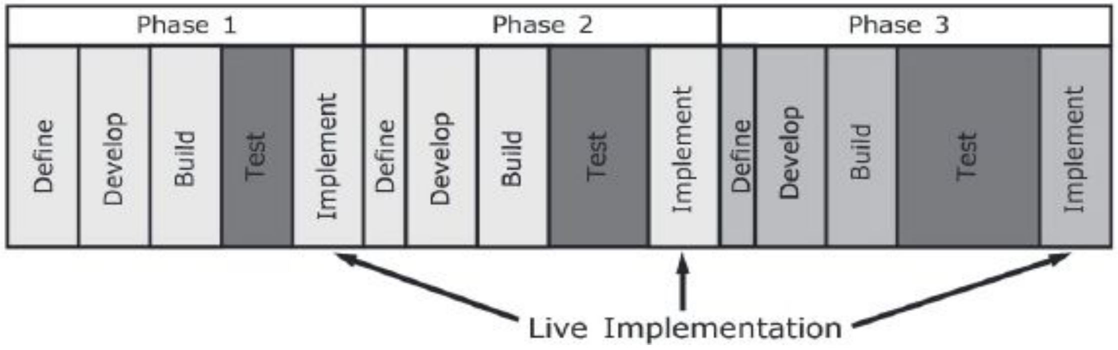
2.4 Software Development and Software Testing ctd.

A common type of V-model uses four (4) **test levels.** Following are the test levels and there objectives

- **Component/Unit testing**: searches for defects in and verifies the functioning of software components that are separately testable.
- **Integration testing**: Test interfaces between components, interactions to different parts of a system.
- **System testing**: concerned with the behavior of the whole system/product as defined by the scope of a development project. Verification against specified requirements is the main focus.
- **Acceptance testing**: validation testing with respect to user needs, requirements and business processes conducted to determine whether or not to accept the system.

2.4 Software Development and Software Testing ctd.

Iterative and Incremental Development Models



Source: Foundations of Software Testing ISTQB Certification,Rex Black, Erik van Veenendaal, Dorothy Graham (page 39)

- Most of the test tasks in sequential model happens in an iterative and incremental life cycle, but their timing and extent may vary.
- The testing tasks may be undertaken in a different order and not necessarily sequentially.
- There are likely to be fewer entry and exit criteria between activities compared with sequential models.

2.4 Software Development and Software Testing ctd.

Common issues with iterative and incremental model include:

- More regression testing.
- Defects outside the scope of the iteration or increment.
- Less thorough testing.

Examples of iterative and incremental development models,

- Rational Unified Process (RUP)
- Scrum
- Kanban
- Spiral (or prototyping)
- Agile Development
- Extreme Programming (XP)

2.4 Software Development and Software Testing ctd.

- Proponents of the Scrum and XP approaches emphasize testing throughout the process.
- Each iteration(sprint) culminates in a short period of testing, often with an independent tester as well as a business representative.
- Developers are to write and run test cases for their code, and leading practitioners use tools to automate those tests and to measure structural coverage of the tests.
- Agile development provides both **benefits** and **challenges** for testers.

2.4 Software Development and Software Testing ctd.

Some of the benefits are:

- The focus on working software and good quality code.
- Simplicity of design that should be easier to test.
- The inclusion of testing as part of and the starting point of software development (test-driven development).

Some of the challenges are:

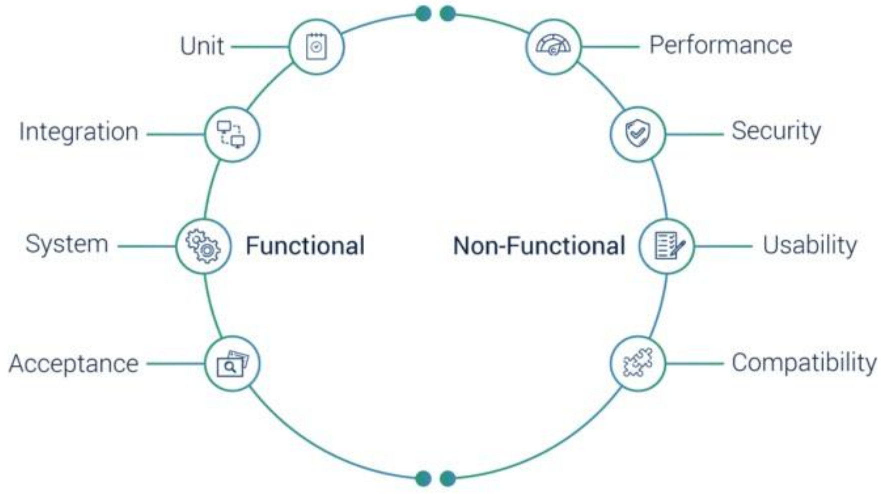
- A constant time pressure and less time to think about the testing for new features.
- Regression testing become extremely important.
- Test designing is less formal and subject to change.
- Testers may be acting more as coaches in testing to both stakeholders and developers.

2.4 Software Development and Software Testing ctd.

Software Development life cycle models in context

- It is important to choose a development model that is suitable for your own situation or context.
- The context also determines the test levels and/or test activities

2.5 Software Testing Methodologies



2.5 Software Testing Methodologies

White – box testing

- Also referred to as
  - Structural testing
  - Glass-box testing
  - Clear – box testing
- Testing based on an analysis of the internal structure – based of the component or system.
- Analyze the internal structures the used data structures, internal design, code structure and the working of the software.
- In white box testing, code is visible to testers.

2.5 Software Testing Methodologies ctd.

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object, and function on an individual basis

2.5 Software Testing Methodologies ctd.

Important White – box testing techniques.

- Statement Coverage
  - In this technique, the aim is to traverse all statement at least once. Hence, each line of code is tested
- Decision Coverage
  - Technique which reports the true or false outcomes of each Boolean expression of the source code
- Branch Coverage
  - In this technique, test cases are designed so that each branch from all decision points are traversed at least once.
- Condition Coverage
  - In this technique, all individual conditions must be covered.

Techniques continued to next slide

2.5 Software Testing Methodologies ctd.

- Multiple Condition Coverage
  - In this technique, all the possible combinations of the possible outcomes of conditions are tested at least once.
- Finite State Machine Coverage
  - In this coverage method, you need to look for how many time-specific states are visited, transited. It also checks how many sequences are included in a finite state machine
- Path Coverage
  - This technique corresponds to testing all possible paths which means that each statement and branch is covered.

2.5 Software Testing Methodologies ctd.

Black – box Testing

- Also referred to as Behavioral Testing
- A software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths.
- Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications

2.5 Software Testing Methodologies ctd.

Types of Black – box testing

- Functional Testing
    - This black box testing type is related to the functional requirements of a system; it is done by software testers.
- Non – Functional Testing
    - This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
- Regression Testing
    - Regression testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

2.5 Software Testing Methodologies ctd.

Grey – Box testing

- A software testing technique to test a software product or application with partial knowledge of internal structure of the application.
- The purpose of grey box testing is to search and identify the defects due to improper code structure or improper use of applications.
- In **White Box testing** internal structure (code) is **known**
- In **Black Box testing** internal structure (code) is **unknown**
- In **Grey Box Testing** internal structure (code) is **partially known**

2.5.2 Functional Testing, Non – Functional Testing, Change – Related Testing

“**Function** of a system/component is what it does, which typically described as use cases, user stories, functional specification or business requirement specification.”

Functional Testing

- Testing Conducted to evaluate the compliance of a component or system with functional requirements.
- Functional tests are based on functions, described in documents or understood by the testers.
- May be performed at **all** test level

2.5.2 Functional Testing, Non – Functional Testing, Change – Related Testing ctd.

Functional testing can also be done focusing on following:

- Suitability
- Interoperability testing
- Security
- Accuracy
- Compliance

2.5.2 Functional Testing, Non – Functional Testing, Change – Related Testing ctd.

Two main perspectives of functional testing;

1. Requirement based testing
    - Uses a specification of the functional requirements for the system as the basis for designing tests.
    - Requirements are prioritized based on a risk criteria and used to prioritize the tests. This will ensure that the most important and critical tests are included in the testing effort

2. Business process based testing
    - Uses knowledge of the business processes.
    - Use cases are very useful basis for test cases from a business perspective.

2.5.2 Functional Testing, Non – Functional Testing, Change – Related Testing ctd.

- The techniques used for functional testing are often specification based, but experience – based techniques can also be used.
- The thoroughness of functional testing can be measured by a coverage measure based on elements of the functions that w can list.
- Special skills or knowledge may be needed for functional testing, particularly for specialized application domain such as medical device software.

2.5.2 Functional Testing, Non – Functional Testing, Change – Related Testing ctd.

Non functional testing

- The testing of the quality characteristics, or non – functional attributes of the system. It is a testing of how well the system works.
- Performed at all test levels.
- Non-functional testing includes (but not limited to),
    - Performance testing
        - Used for testing the speed, response time, stability, reliability, scalability and resource usage of a software application under particular workload.
    - Load testing
        - Involves testing the system's loading capacity. Loading capacity means more and more people can work on the system simultaneously.

Continued to next slide

2.5.2 Functional Testing, Non – Functional Testing, Change – Related Testing ctd.

- Stress testing
  - A type of software testing that verifies stability & reliability of software application. The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn’t crash under crunch situations.
- Usability testing
  - A testing method for measuring how easy and user-friendly a software application is
- Portability testing
  - A testing methodology that determines the ease or difficulty with which a software component or an application can be moved from one environment to another.
- Security testing
  - A type of Software Testing that uncovers vulnerabilities, threats, risks in a software application and prevents malicious attacks from intruders.

2.5.2 Functional Testing, Non – Functional Testing, Change – Related Testing ctd.

- The thoroughness of non-functional testing can be measured by the coverage of non – functional elements
- Coverage gaps could be identified if there is traceability between non-functional tests and non – functional requirements.
- Special skills or knowledge may be needed for non – functional testing, such as for performance testing or security testing

2.5.2 Functional Testing, Non – Functional Testing, Change – Related Testing ctd.

Change related testing

“If you have made a change to the software, you will have changed the way it functions and/or its structure.“

- There are two things to be aware of when changes are made:
  - The change it self
  - Any other effects of the change.

1. Confirmation testing (re - testing)

- Dynamic test conducted after fixing defects with the objective to confirm that failures caused by those defects do not occur anymore.
- Important to ensure that steps leading up to the failure are carried out in exactly the same way
- If the test passes now – can compare at least one part of software is correct where the defect was.

2.5.2 Functional Testing, Non – Functional Testing, Change – Related Testing ctd.

But this is not enough,

The fix may have introduced or uncovered a different defect elsewhere in the software.

2. Regression Testing

- Testing of a previously tested component or system following modification to ensure that defects have not been introduced or have been uncovered in unchanged areas of the software as a result of the changes made.
- Regression tests are executed whenever the software changes, either as result of fixes or new or changed functionality.
- It is good idea to execute them when some aspect of the environment changes, e.g. when a new version of the host operating system is introduced.

2.5.2 Functional Testing, Non – Functional Testing, Change – Related Testing ctd.

Both confirmation testing and regression testing are done at all test levels.

2.5.3 Test Types and Test Levels

Test Levels

- Groups of test activities that are organized and managed together.

Each test level has the following clearly identified:

- Specific **test objectives** for the test level.
- The **test basis,** the work product(s) used to derive thee test conditions and **test cases.**
- The **test object**.
- The typical defects and failures that we are looking for at this level.
- Specific approaches and responsibilities for this test level.
- The **test environment.**

2.5.3 Test Types and Test Levels ctd.

Test Levels:

- Component Testing (Module testing, unit testing)
  - The testing of individual hardware or software components
- Integration Testing
  - Component integration testing
    - Testing performed to expose defects in the interfaces and interactions between integrated components.
  - System integration testing
    - Testing the combination and interaction of systems.
- System Testing
  - Testing an integrated system to verify that it meets specified requirements. (In practice system testing is about both verification and validation).

2.5.3 Test Types and Test Levels ctd.

- Acceptance testing
  - Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to **accept** the system

Different forms of acceptance testing

- User acceptance testing (UAT)
  - Acceptance testing conducted in a real or simulated operational environment
  - by intended users
  - focusing on their needs, requirements and business process.

2.5.3 Test Types and Test Levels ctd.

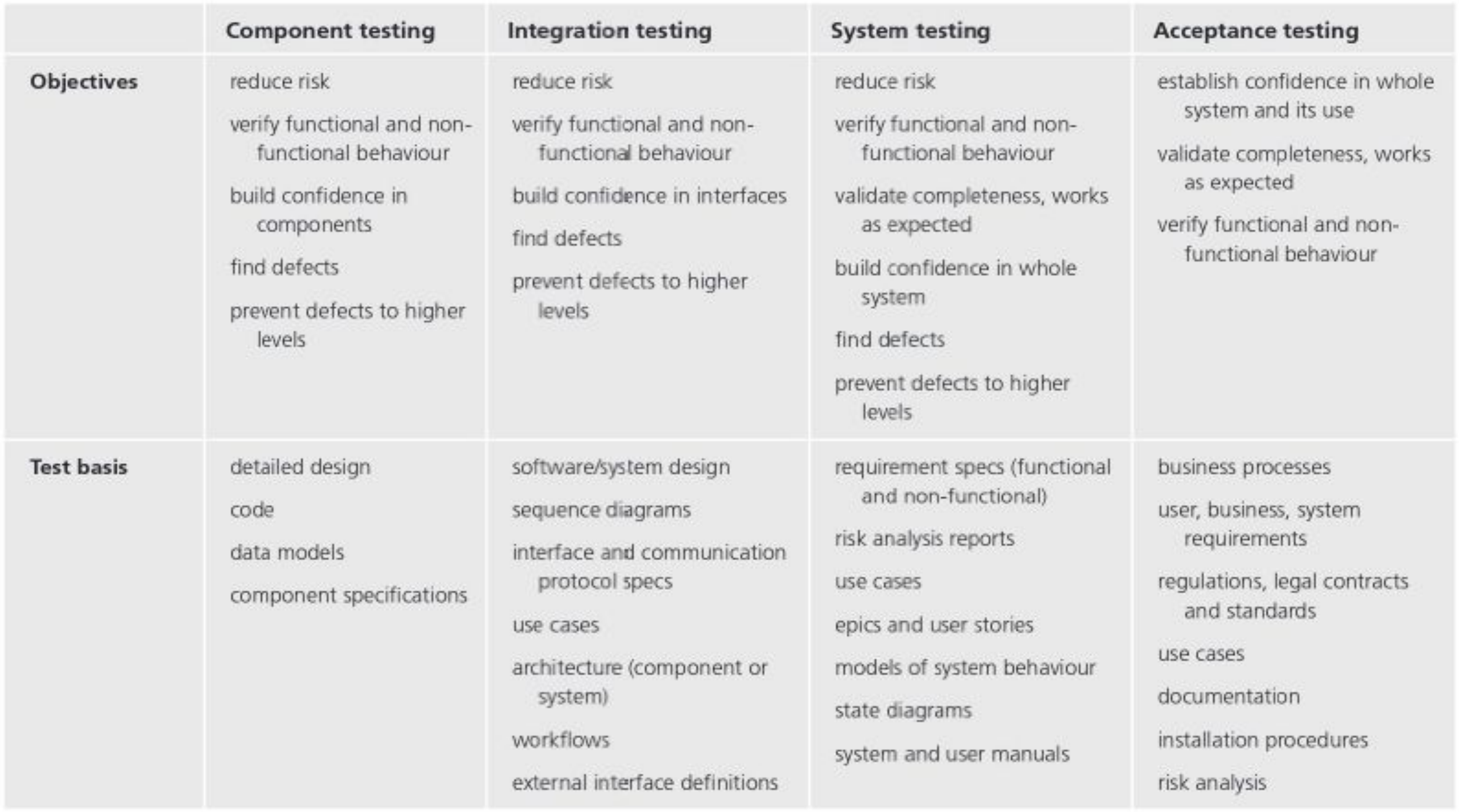
- Operational Acceptance testing (OAT)
  - Typically performed in a (simulated) operational environment
  - By operations and/or systems administration staff
  - Focusing on operational aspects
- Contractual Acceptance testing
  - Acceptance testing conducted to verify whether a system satisfies contractual requirements.
- Regulatory acceptance testing
  - Acceptance testing conducted to verify whether a system conforms to relevant laws, policies and regulations.
- Alpha Testing
  - Simulated or actual operational testing conducted in the developers test environment
  - By roles outside the development organization.

2.5.3 Test Types and Test Levels ctd.

- Beta testing
  - Simulated or actual operational testing conducted at an external site
  - By roles outside the development organization.

2.5.3 Test Types and Test Levels ctd.

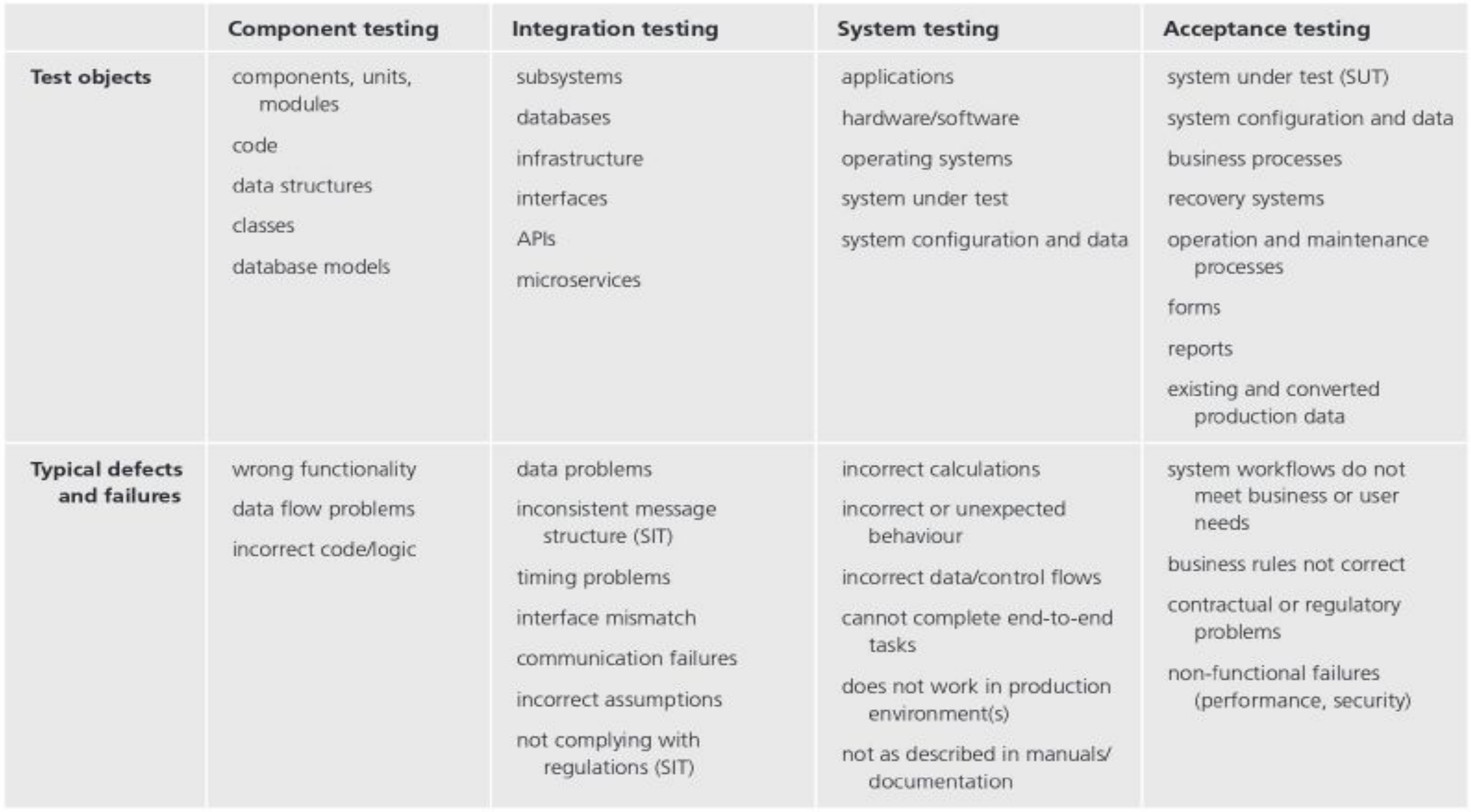
Test level characteristics: summary



	Component testing	Integration testing	System testing	Acceptance testing
Objectives	reduce risk verify functional and non-functional behaviour build confidence in components find defects prevent defects to higher levels	reduce risk verify functional and non-functional behaviour build confidence in interfaces find defects prevent defects to higher levels	reduce risk verify functional and non-functional behaviour validate completeness, works as expected build confidence in whole system find defects prevent defects to higher levels	establish confidence in whole system and its use validate completeness, works as expected verify functional and non-functional behaviour
Test basis	detailed design code data models component specifications	software/system design sequence diagrams interface and communication protocol specs use cases architecture (component or system) workflows external interface definitions	requirement specs (functional and non-functional) risk analysis reports use cases epics and user stories models of system behaviour state diagrams system and user manuals	business processes user, business, system requirements regulations, legal contracts and standards use cases documentation installation procedures risk analysis

Source: Foundations of Software Testing ISTQB Certification,Rex Black, Erik van Veenendaal, Dorothy Graham (page 60)

2.5.3 Test Types and Test Levels ctd.



	Component testing	Integration testing	System testing	Acceptance testing
Test objects	components, units, modules code data structures classes database models	subsystems databases infrastructure interfaces APIs microservices	applications hardware/software operating systems system under test system configuration and data	system under test (SUT) system configuration and data business processes recovery systems operation and maintenance processes forms reports existing and converted production data
Typical defects and failures	wrong functionality data flow problems incorrect code/logic	data problems inconsistent message structure (SIT) timing problems interface mismatch communication failures incorrect assumptions not complying with regulations (SIT)	incorrect calculations incorrect or unexpected behaviour incorrect data/control flows cannot complete end-to-end tasks does not work in production environment(s) not as described in manuals/ documentation	system workflows do not meet business or user needs business rules not correct contractual or regulatory problems non-functional failures (performance, security)

Source: Foundations of Software Testing ISTQB Certification,Rex Black, Erik van Veenendaal, Dorothy Graham (page 61)

2.5.3 Test Types and Test Levels ctd.

- Each test type is applicable at every test level.

- **Example** – Consider a banking application
  - There are multiple features available, Some are visible to user and others are behind the scenes.
  - Functional tests at each level
    - Component testing – how the components should calculate compound interest.
    - System testing – how account holders can apply for line of credit.
    - Acceptance testing – how the banker handles approving or declining credit application.

Example Continued to next slide

2.5.3 Test Types and Test Levels ctd.

- Non – functional tests at each level
    - Component testing – the time or number of CPU cycles perform a complex interest calculation.
    - System testing – portability tests on whether the presentation layer works on supported browser and mobile device.
    - Acceptance testing – usability tests for accessibility of banker’s credit processing interface for people with disabilities.
- White-box tests at each level
    - Component testing – achieve 100% statement and decision coverage for all components performing financial calculations.
    - System testing – coverage of sequences of all possible inquiry types sent to the credit score microservice
    - Acceptance testing – coverage of all supported financial data file structures and value ranges for bank-to-bank transfers.

2.6 Static vs Dynamic Testing

Image source : https://www.softwaretestingmaterial.com/wp-content/uploads/2021/09/Static-Testing-vs-Dynamic-Testing

2.6 Static vs Dynamic Testing

Static Testing

- a type of software testing in which software application is tested without code execution.
- Manual or automated reviews of code, requirement documents and document design are done in order to find the errors.
- The main objective of static testing is to improve the quality of software applications by finding errors in early stages of software development process.

Static Testing Techniques

- Informal reviews
- Technical reviews
- Walkthrough
- Inspection
- Static Code review

2.6 Static vs Dynamic Testing ctd.

Dynamic Testing

- a code is executed.
- It checks for functional behavior of software system, memory/cpu usage and overall performance of the system.
- The main objective of this testing is to confirm that the software product works in conformance with the business requirements.

Dynamic Testing Techniques

- Unit testing
- Integration Testing
- System Testing

2.6 Static vs Dynamic Testing ctd.

Key Differences

- Static testing was done without executing the program whereas Dynamic testing is done by executing the program.
- Static testing is about the prevention of defects whereas Dynamic testing is about finding and fixing the defects.
- Static testing does the verification process while Dynamic testing does the validation process.
- Static testing is performed before compilation whereas Dynamic testing is performed after compilation.

Static testing techniques are structural and statement coverage while Dynamic testing techniques are Boundary Value Analysis & Equivalence Partitioning.

Summary

Testing Process	• There is no “one size fits all” test process
Test Activities	• The activities in the process may overlap or take place concurrently or iteratively
Software Testing Methodologies	• White – box testing vs Black – box testing vs Grey – box testing • Functional vs Non- Functional testing