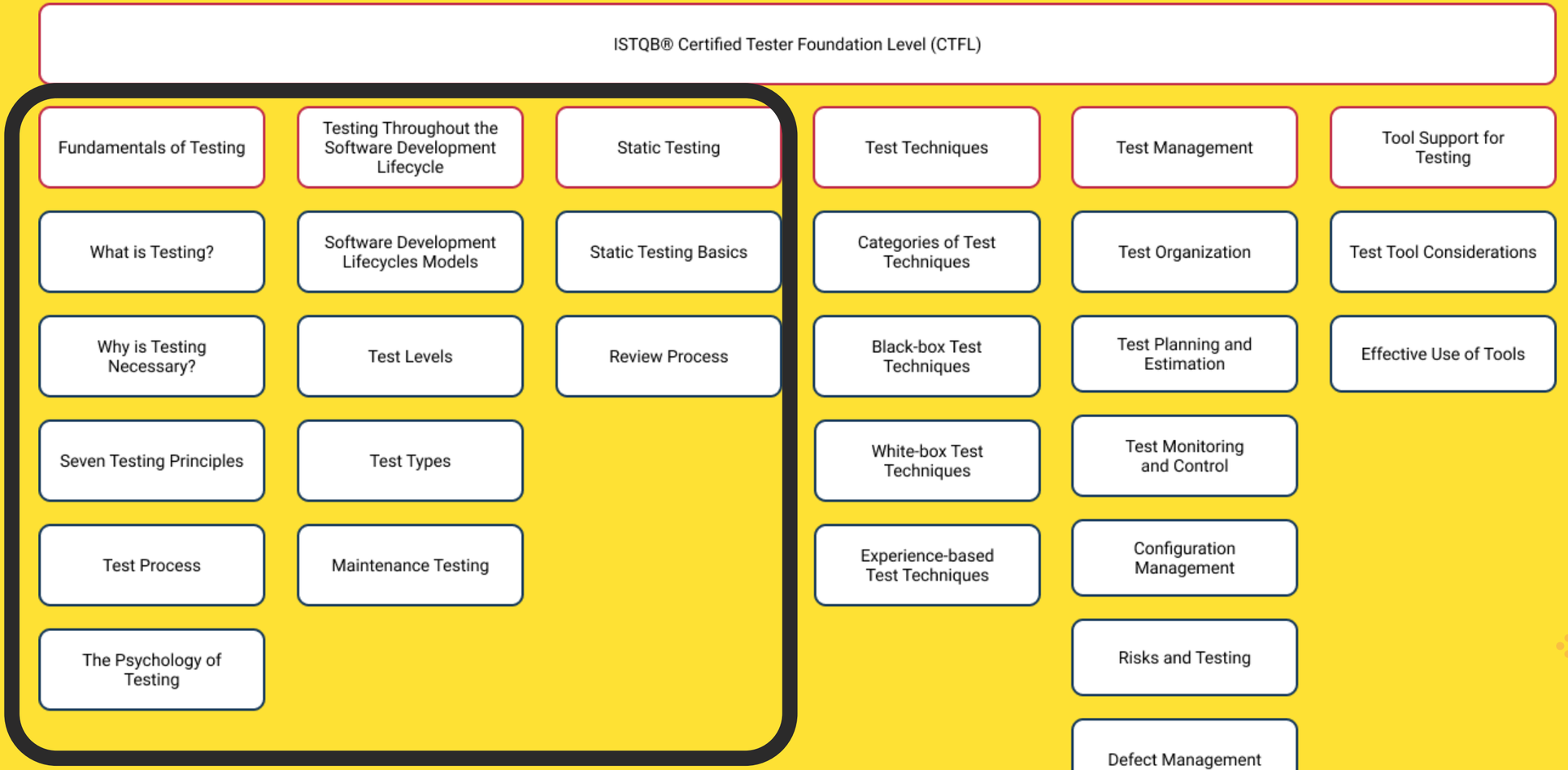
ISTQB-CTFL

SOFTWARE QUALITY CONTROL & ASSURANCE

Mae Kristine Clor

ISQTB-CTFL



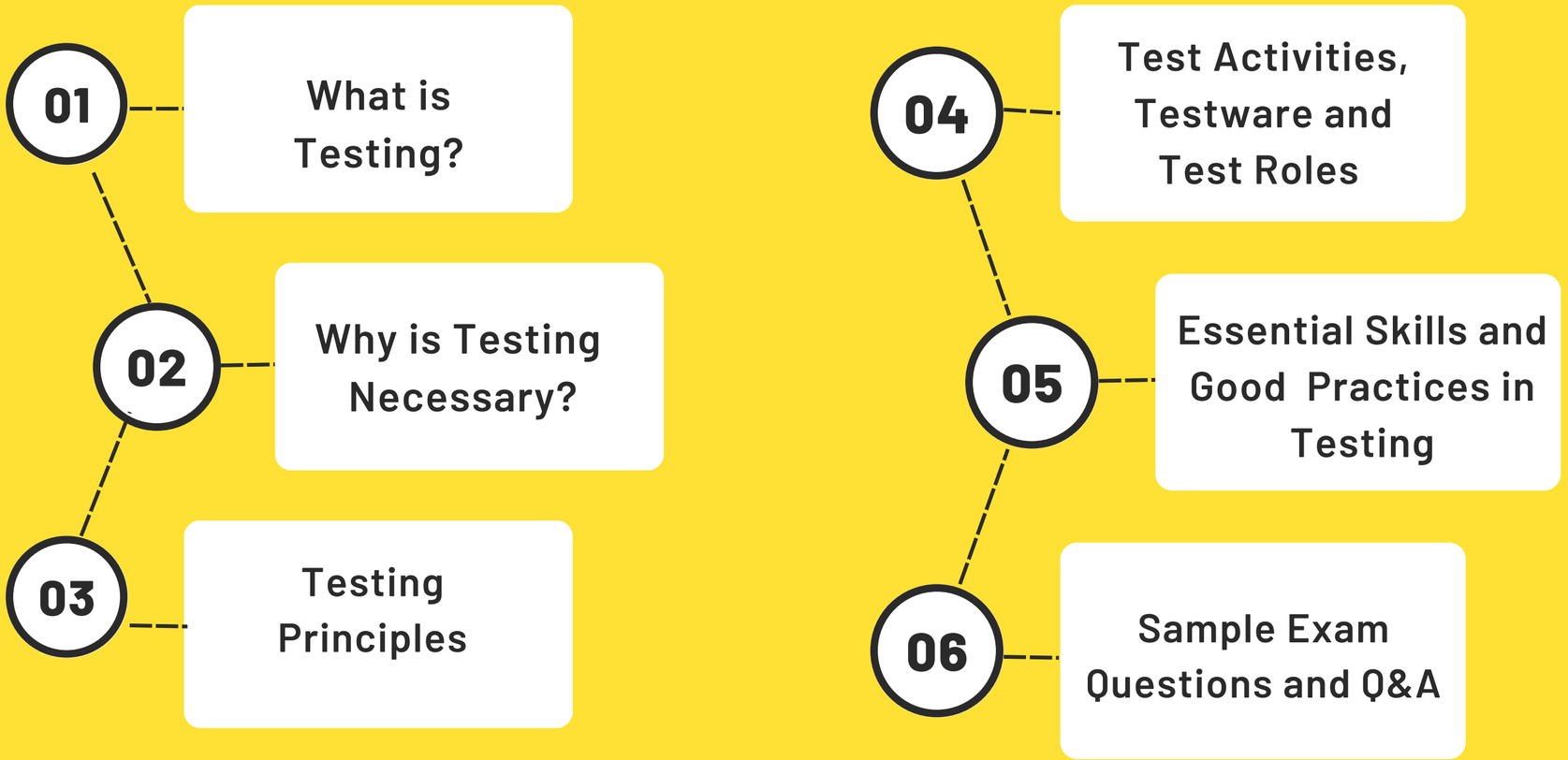
ISTQB® is the leading global certification scheme in the field of software testing

Core Foundation - gives practical knowledge of the fundamental concepts of software testing. It is the **prerequisite** to the other modules within the scheme which offer depth and specialization.

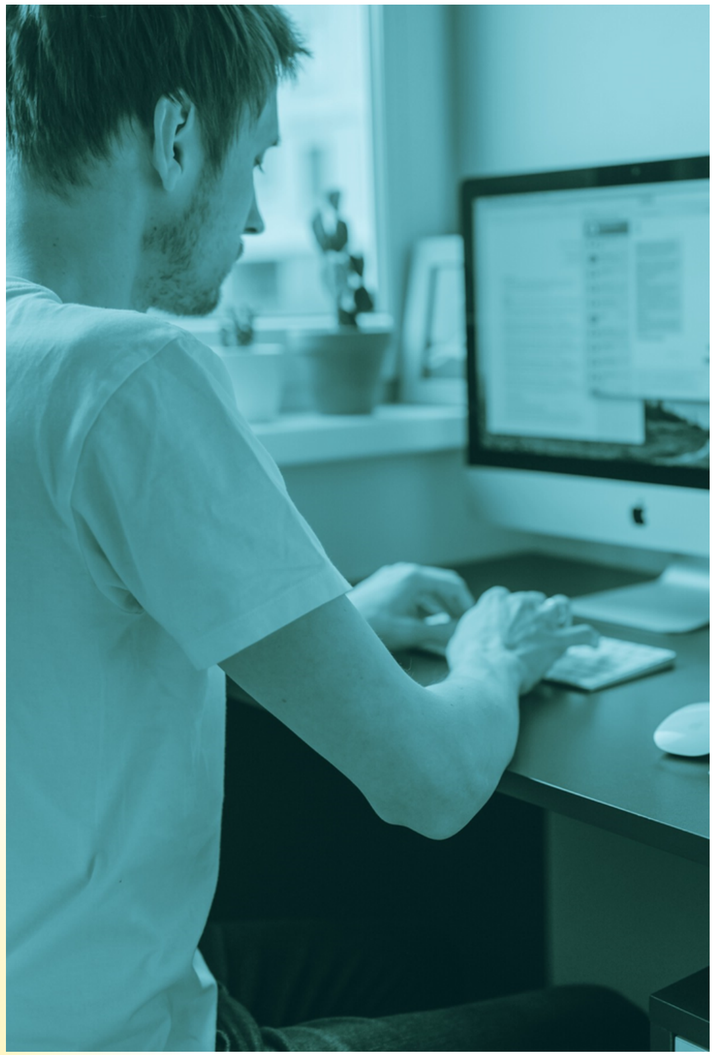
ISTQB®
International Software Testing Qualifications Board

ISTQB_CTFL_SYLLABUS-V4.0 (2023)

FUNDAMENTALS OF TESTING



WHAT IS TESTING?



IS A WAY TO ASSESS THE SOFTWARE'S QUALITY AND REDUCE SOFTWARE FAILURE IN OPERATION

Software testing is a set of activities to discover defects and evaluate the quality of software artifacts. These artifacts, when being tested, are known as test objects

OBJECTIVES OF TESTING

1. Evaluating work products such as requirements, user stories, designs, and code
2. Triggering failures and finding defects
3. Ensuring required coverage of a test object
4. Reducing the level of risk of inadequate software quality
5. Verifying whether specified requirements have been fulfilled
6. Verifying that a test object complies with contractual, legal, and regulatory requirements
7. Providing information to stakeholders to allow them to make informed decisions
8. Building confidence in the quality of the test object
9. Validating whether the test object is complete and works as expected by the stakeholders

CATEGORIES OF TESTING

Dynamic

- **Involves the execution** of the component or system being tested
- Functional, regression, integration, performance, etc.

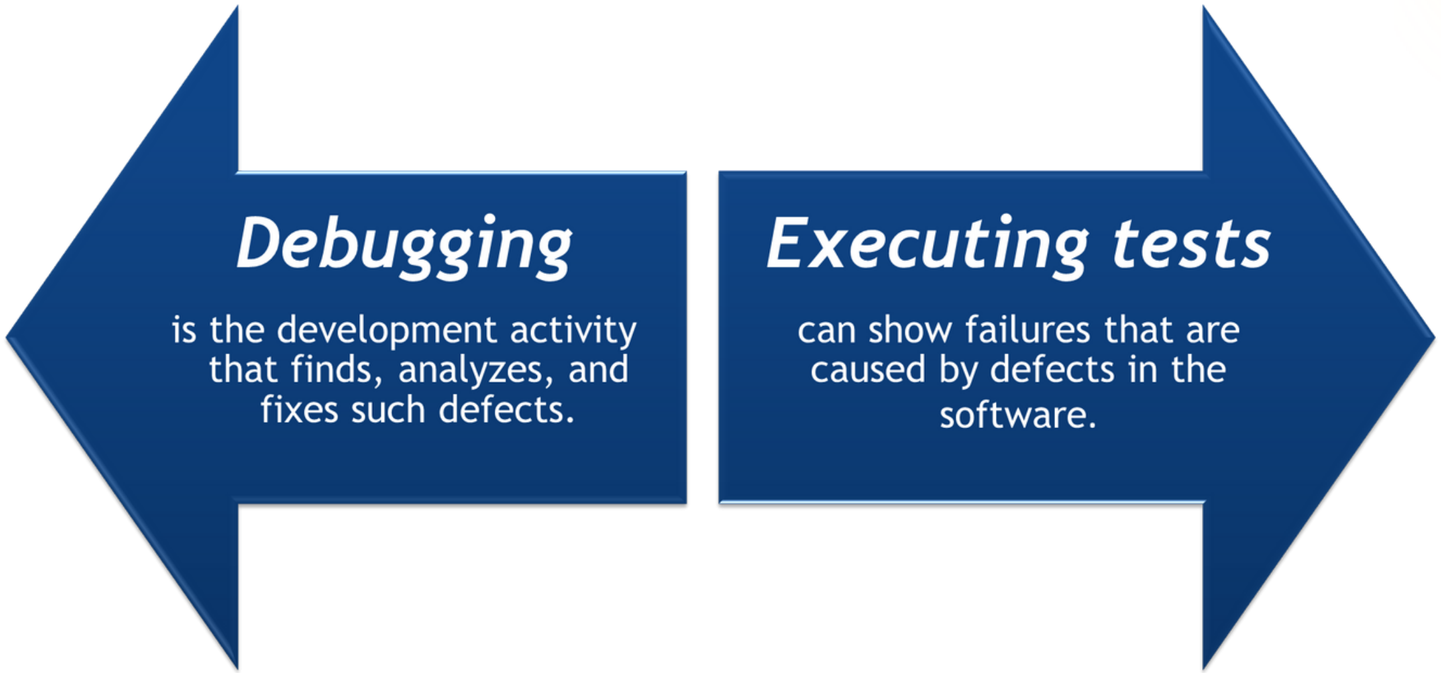
Static

- **Does not involve the execution** of the component or system being tested
- Reviewing work products such as requirements, user stories, and source code

TESTING VS. DEBUGGING

Testing and Debugging

Testers are responsible for the initial test and the final confirmation test, while **developers** do the debugging, associated component integration testing



The typical debugging process in this case involves:

- Reproduction of a failure
- Diagnosis (finding the root cause)
- Fixing the cause

Subsequent confirmation testing checks whether the fixes resolved the problem.
Preferably, confirmation testing is done by the same person who performed the initial test

WHY IS TESTING NECESSARY?

Testing's Contributions to Succes

- Testing provides a cost-effective means of detecting defects.
- Testing provides a means of directly evaluating the quality of a test object at various stages in the SDLC
- Testing provides users with indirect representation on the development project
- Testing may also be required to meet contractual or legal requirements

Testing and Quality Assurance (QA)

QC is a product-oriented, corrective approach that focuses on those activities supporting the achievement of appropriate levels of quality

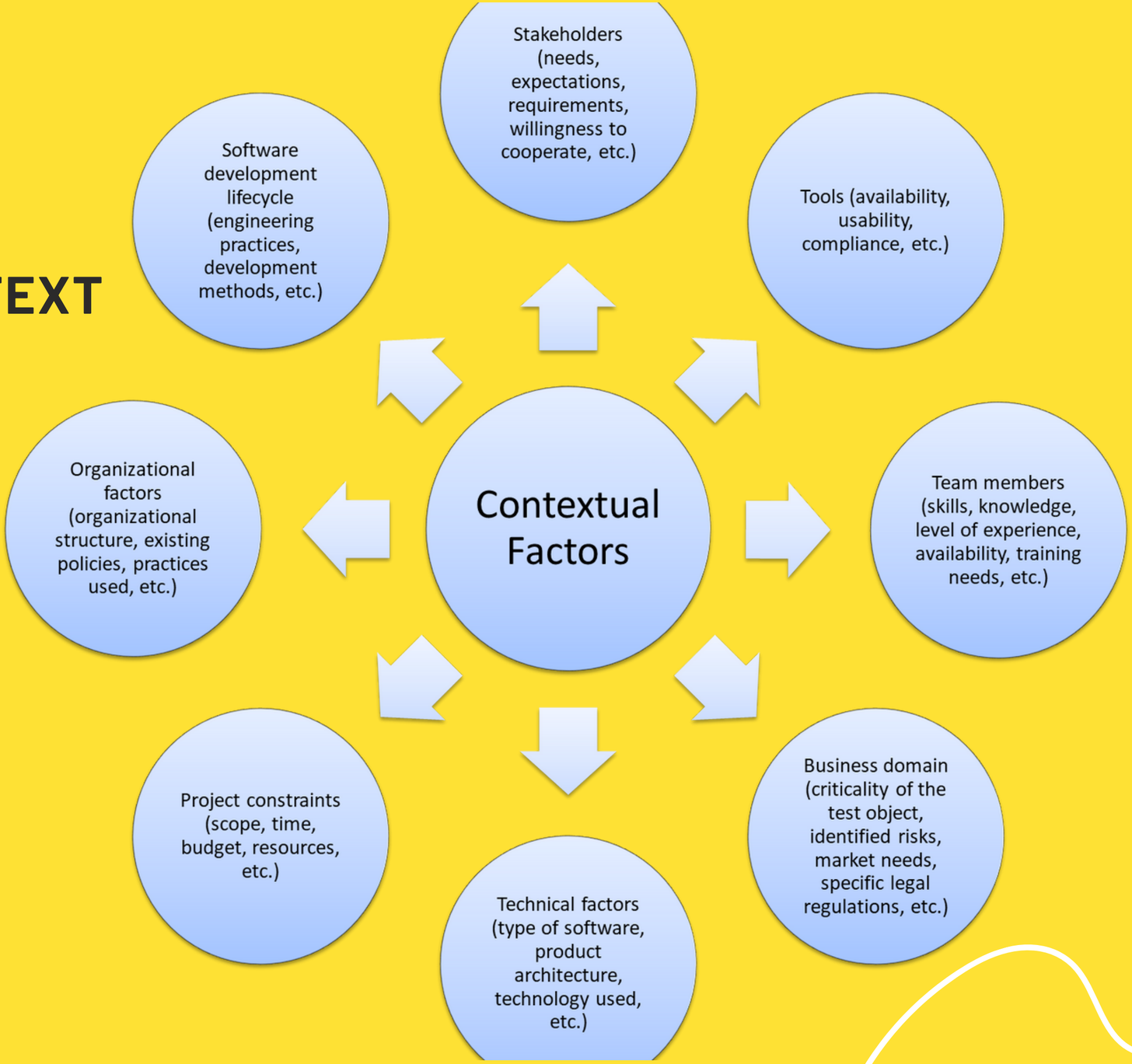
QA is a process-oriented, preventive approach that focuses on the implementation and improvement of processes

Test results are used by QA and QC. In QC they are used to fix defects, while in QA they provide feedback on how well the development and test processes are performing.

SEVEN TESTING PRINCIPLES

1. Testing shows the presence of defects, not their absence	2. Exhaustive testing is impossible	3. Early testing saves time and money	4. Defects cluster together	5. Beware of the pesticide paradox	6. Testing is context dependent	7. Absence-of-errors is a fallacy
				Tests wear out		
Testing can show that defects are present but cannot prove that there are no defects.	Testing everything (all combinations of inputs and preconditions) is not feasible	Early testing is sometimes referred to as shift left. Testing early in the software development lifecycle helps reduce or eliminate costly changes	A small number of modules usually contains most of the defects discovered during pre-release testing, or is responsible for most of the operational failures.	If the same tests are repeated over and over again, eventually these tests no longer find any new defects.	Testing is done differently in different contexts.	Some organizations expect that testers can run all possible tests and find all possible defects, but principles 2 and 1, respectively, tell us that this is impossible.

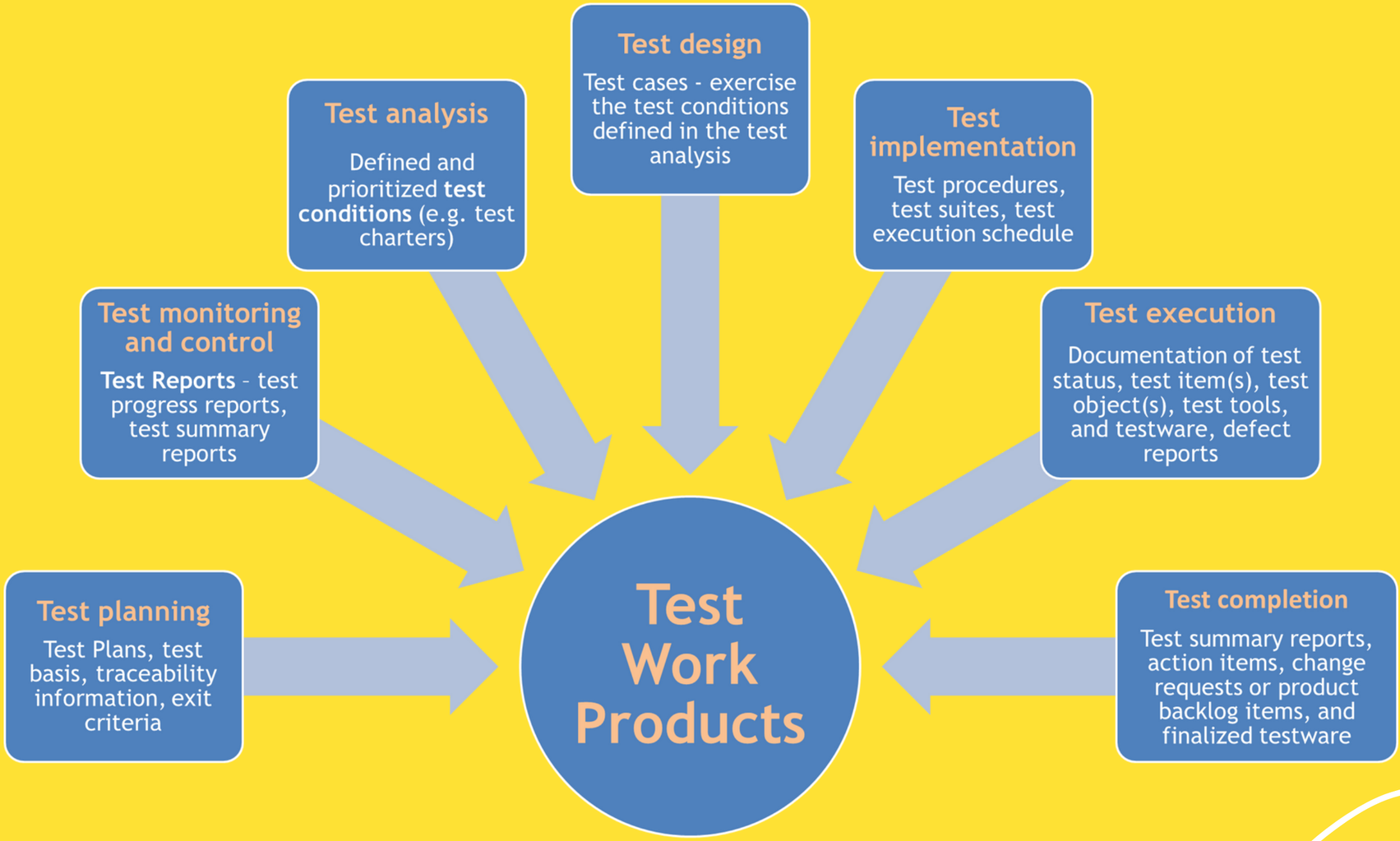
TEST PROCESS IN CONTEXT



TEST ACTIVITIES AND TASKS

Test planning	Activities that define the objectives of testing and the approach for meeting test objectives
Test monitoring and control	Comparison of actual progress against planned progress
Test analysis	Test analysis determines “what to test” in terms of measurable coverage criteria
Test design	Answers the question “how to test?”
Test implementation	Answers the question “do we now have everything in place to run the tests?”
Test execution	Test suites are run in accordance with the test execution schedule
Test completion	Collect data from completed test activities to consolidate experience, testware, and any other relevant information

TEST WORK PRODUCTS



TRACEABILITY BETWEEN THE TEST BASIS AND TEST WORK PRODUCTS

Good traceability supports:

- Analyzing the impact of changes
- Making testing auditable
- Meeting IT governance criteria
- Improving the understandability of test progress reports and test summary reports
- 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

ESSENTIAL SKILLS AND GOOD PRACTICES IN TESTING

Generic Skills Required for Testing

- Testing knowledge (to increase the effectiveness of testing, e.g., by using test techniques)
- Good communication skills, active listening, being a team player
- Technical knowledge (to increase the efficiency of testing, e.g., by using appropriate test tools)
- Thoroughness, carefulness, curiosity, attention to detail, being methodical (to identify defects, especially the ones that are difficult to find)
- Analytical thinking, critical thinking, and creativity (to increase the effectiveness of testing)
- Domain knowledge (to be able to understand and to communicate with end users/business representatives)

Whole Team Approach

In the whole-team approach any team member with the necessary knowledge and skills can perform any task, and everyone is responsible for quality.

Testers work closely with other team members to ensure that the desired quality levels are achieved.

INDEPENDENCE OF TESTING

- Work products can be tested by their author (no independence)
- by the author's peers from the same team (some independence)
- by testers from outside the author's team but within the organization (high independence)
- by testers from outside the organization (very high independence)

Benefit

Independent testers are likely to recognize different kinds of failures and defects compared to developers because of their different backgrounds, technical perspectives, and biases.

Drawback

Independent testers may be isolated from the development team. Developers may lose a sense of responsibility for quality. Independent testers may be seen as a bottleneck or blamed for release delays.

TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFECYCLE

01 Testing in the Context of a Software Development Lifecycle

02 Test Levels and Test Types

03 Maintenance Testing

04 Sample Exam Questions and Q&A

TESTING IN THE CONTEXT OF A SOFTWARE DEVELOPMENT LIFECYCLE

In any software development lifecycle model, 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

Test analysis and design for a given test level begin during the corresponding development activity

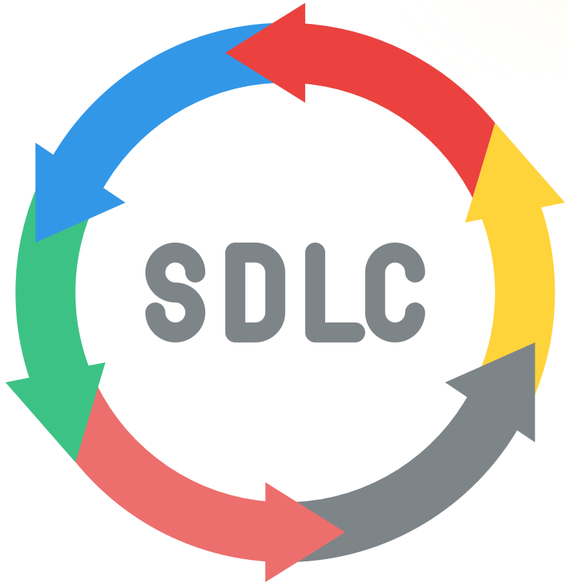
Testers participate in discussions to define and refine requirements and design, and are involved in reviewing work products

Impact of the Software Development Lifecycle on Testing

- Scope and timing of test activities (e.g., test levels and test types)
- Level of detail of test documentation
- Choice of test techniques and test approach
- Extent of test automation
- Role and responsibilities of a tester

SOFTWARE DEVELOPMENT LIFECYCLE MODELS

SEQUENTIAL	describes the software development process as a linear, sequential flow of activities (Waterfall, V-Model)
INCREMENTAL	integrates the test process throughout the development process, implementing the principle of early testing.
ITERATIVE	groups of features are specified, designed, built, and tested together in a series of cycles, often of a fixed duration. (Rational Unified Process, Scrum, Kanban, Spiral)



Software development lifecycle models must be selected and adapted to the **context** of project and product characteristics. An appropriate software development lifecycle model should be selected and adapted based on the **project goal, the type of product being developed, business priorities**.

TESTING AS A DRIVER FOR SOFTWARE DEVELOPMENT

Test-Driven Development (TDD):

- Directs the coding through test cases (instead of extensive software design) (Beck 2003)
- Tests are written first, then the code is written to satisfy the tests, and then the tests and code are refactored

Acceptance Test-Driven Development (ATDD) (see section 4.5.3):

- Derives tests from acceptance criteria as part of the system design process (Gärtner 2011)
- Tests are written before the part of the application is developed to satisfy the tests

Behavior-Driven Development (BDD):

- Expresses the desired behavior of an application with test cases written in a simple form of natural language, which is easy to understand by stakeholders – usually using the Given/When/Then format. (Chelimsky 2010)
- Test cases are then automatically translated into executable tests

DEVOPS AND TESTING

DevOps is an organizational approach aiming to create synergy by getting development (including testing) and operations to work together to achieve a set of common goals.

RETROSPECTIVES AND PROCESS IMPROVEMENT

What was successful, and should be retained?

What was not successful and could be improved?

How to incorporate the improvements and retain the successes in the future?

Typical benefits for testing include:

- Increased test effectiveness/efficiency (e.g., by implementing suggestions for process improvement)
- Increased quality of testware (e.g., by jointly reviewing the test processes)
- Team bonding and learning (e.g., as a result of the opportunity to raise issues and propose improvement points)
- Improved quality of the test basis (e.g., as deficiencies in the extent and quality of the requirements could be addressed and solved)
- Better cooperation between development and testing (e.g., as collaboration is reviewed and optimized regularly)

SHIFT-LEFT APPROACH

The principle of early testing. Shift-left normally suggests that testing should be done earlier

How to achieve "shift left":

- Reviewing the specification from the perspective of
- Writing test cases before the code is written
- Using CI and even better CD as it comes with fast feedback and automated component tests
- Completing static analysis of source code prior to dynamic testing, or as part of an automated process
- Performing non-functional testing starting at the component test level, where possible

TEST LEVELS

Component testing (also known as unit or module testing) focuses on components that are separately testable.

Objectives of component testing include:

- Reducing risk
- Verifying whether the functional and non-functional behaviors of the component are as designed and specified
- Building confidence in the component's quality
- Finding defects in the component
- Preventing defects from escaping to higher test levels

Test Basis	Test objects	Typical defects and failures
• Detailed design • Code • Data model • Component specifications	• Components, units or modules • Code and data structures • Classes • Database modules	• Incorrect functionality • Data flow problems • Incorrect code and logic

TEST LEVELS

Integration testing focuses on interactions between components or systems.

Objectives of integration testing include:

- Reducing risk
- Verifying whether the functional and non-functional behaviors of the interfaces are as designed and specified
- Building confidence in the quality of the interfaces
- Finding defects (which may be in the interfaces themselves or within the components or systems)
- Preventing defects from escaping to higher test levels

Test Basis	Test objects	Typical defects and failures
• System and software requirement specifications (functional and non-functional) • Risk analysis reports • Use cases • Epics and user stories • Models of system behavior • State diagrams • System and user manuals	• Applications • Hardware/software systems • Operating systems • System under test (SUT) • System configuration and configuration data	• Incorrect calculations • Incorrect or unexpected system functional or non-functional behavior • Incorrect control and/or data flows within the system • Failure to properly and completely carry out end-to-end functional tasks • Failure of the system to work properly in the system environment(s) • Failure of the system to work as described in the system and user manuals

➢ **Component integration testing** focuses on the interactions and interfaces between integrated components. Component integration testing is performed after component testing, and is generally automated.

➢ **System integration testing** focuses on the interactions and interfaces between systems, packages, and microservices. System integration testing can also cover interactions with, and interfaces provided by, external organizations (e.g., web services).

TEST LEVELS

System testing focuses on the behavior and capabilities of a whole system or product.

Objectives of system testing include:

•Reducing risk
•Verifying whether the functional and non-functional behaviors of the system are as designed and specified
•Validating that the system is complete and will work as expected
•Building confidence in the quality of the system as a whole
•Finding defects
•Preventing defects from escaping to higher test levels or production

Test Basis	Test objects	Typical defects and failures
• System and software requirement specifications (functional and non-functional) • • Risk analysis reports • • Use cases • • Epics and user stories • • Models of system behavior • • State diagrams • • System and user manuals	• Subsystems • Databases • Infrastructure • Interfaces • APIs • Microservices	• Incorrect data, missing data, or incorrect data encoding • Incorrect sequencing or timing of interface calls • Interface mismatch • Failures in communication between components • Unhandled or improperly handled communication failures between components • Incorrect assumptions about the meaning, units, or boundaries of the data being passed • Inconsistent message structures between systems

TEST LEVELS

Acceptance testing, may produce information to assess the system's readiness for deployment and use by the customer (end-user).

Objectives of acceptance testing include:

•Establishing confidence in the quality of the system as a whole
•Validating that the system is complete and will work as expected
•Verifying that functional and non-functional behaviors of the system are as specified

Test Basis	Test objects	Typical defects and failures
• Business processes • User or business requirements • Regulations, legal contracts, and standards • Use cases and/or user stories • System requirements • System or user documentation • Installation procedures • Risk analysis reports	• System under test • System configuration and configuration data • Business processes for a fully integrated system • Recovery systems and hot sites (for business continuity and disaster recovery testing) • Operational and maintenance processes • Forms • Reports • Existing and converted production data	• 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

TEST LEVELS

Common forms of acceptance testing

- **User acceptance testing (UAT)** of the system is typically focused on validating the fitness for use of the system by intended users in a real or simulated operational environment.
- **Operational acceptance testing (OAT)** of the system by operations or systems administration staff is usually performed in a (simulated) production environment.
- **Contractual and regulatory acceptance testing** is performed against a contract's acceptance criteria for producing custom-developed software. Often performed by users or by independent testers.
- **Alpha and beta testing** are typically used by developers of commercial off-the-shelf (COTS) software who want to get feedback from potential or existing users, customers, and/or operators before the software product is put on the market.
  - Alpha testing is performed at the developing organization's site,
  - Beta testing is performed by potential or existing customers, and/or operators at their own locations.

TEST TYPES

A test type is a group of test activities aimed at testing specific characteristics of a software system, or a part of a system, based on specific test objectives.

Functional Testing

- Involves tests that evaluate functions that the system should perform.
- Functional requirements may be described in work products.
- The functions are “what” the system should do.
- Functional tests should be performed at all test levels
- Considers the behavior of the software (black-box techniques)

Non-functional Testing

- Evaluates characteristics of systems and software such as usability, performance efficiency or security.
- Testing of “how well” the system behaves.
- should be performed at all test levels and done as early as possible.
- Black-box techniques may be used to derive test conditions and test cases

White-box Testing

- Based on the system’s internal structure or implementation - architecture, work flows, and/or data flows within the system
- Measured through structural coverage which is the extent to which some type of structural element has been exercised by tests
- At the component testing level, code coverage is based on the percentage of component code that has been tested, and may be measured in terms of different aspects of code

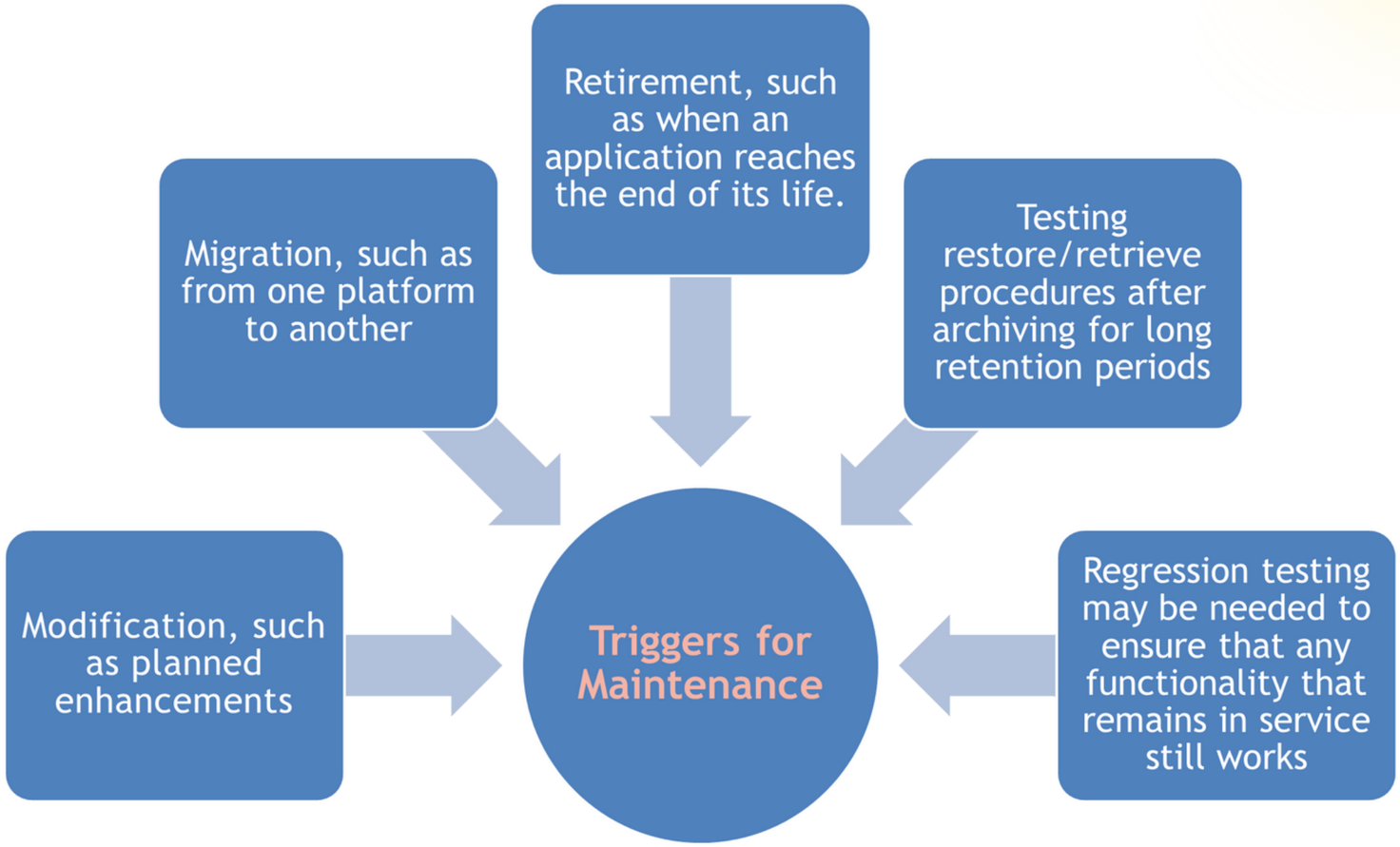
Change-related Testing

- Confirmation testing: Software is tested with all test cases that failed due to the defect, which should be re-executed on the new software version.
- Regression testing involves running tests to detect unintended side-effects.

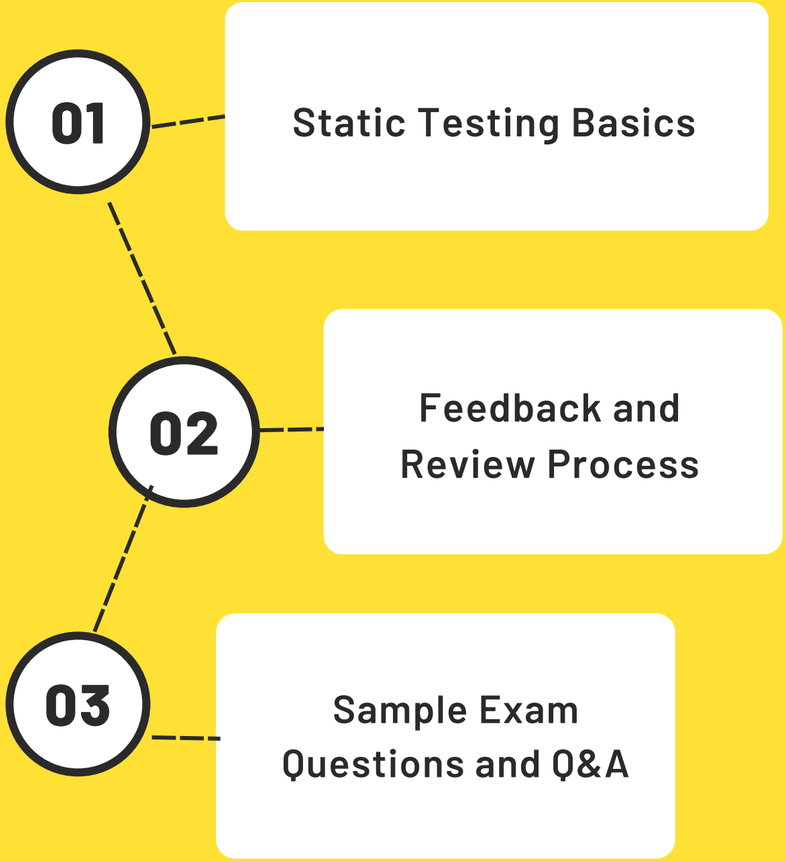
MAINTENANCE TESTING

The scope of maintenance testing depends on:

- The degree of risk of the change
- The size of the existing system
- The size of the change



STATIC TESTING



STATIC TESTING BASICS

Static testing relies on the manual examination of work products

Work Products that Can Be Examined by Static Testing

- Specifications, including business requirements, functional requirements, and security requirements
- Epics, user stories, and acceptance criteria
- Architecture and design specifications
- Code
- Testware, including test plans, test cases, test procedures, and automated test scripts
- User guides
- Web pages
- Contracts, project plans, schedules, and budget planning
- Configuration set up and infrastructure set up
- Models, such as activity diagrams

Value of Static Testing

Static testing can detect defects in the earliest phases of the SDLC, fulfilling the principle of early testing

Static testing provides the ability to evaluate the quality of, and to build confidence in work products.

Even though reviews can be costly to implement, the overall project costs are usually much lower than when no reviews are performed because less time and effort needs to be spent on fixing defects later in the project.

Code defects can be detected using static analysis more efficiently than in dynamic testing, usually resulting in both fewer code defects and a lower overall development effort.

DIFFERENCES BETWEEN STATIC AND DYNAMIC TESTING

- One main distinction is that static testing finds defects in work products directly rather than identifying failures caused by defects when the software is run.
- Another distinction is that static testing can be used to improve the consistency and internal quality of work products, while dynamic testing typically focuses on externally visible behaviors.

Compared with dynamic testing, typical defects that are easier and cheaper to find and fix through static testing include:

- ❑ Requirement defects (e.g., inconsistencies, ambiguities, contradictions, omissions, inaccuracies, and redundancies)
- ❑ Design defects (e.g., inefficient algorithms or database structures, high coupling, low cohesion)
- ❑ Coding defects (e.g., variables with undefined values, variables that are declared but never used, unreachable code, duplicate code)
- ❑ Deviations from standards (e.g., lack of adherence to coding standards)
- ❑ Incorrect interface specifications (e.g., different units of measurement used by the calling system than by the called system)
- ❑ Security vulnerabilities (e.g., susceptibility to buffer overflows)
- ❑ Gaps or inaccuracies in test basis traceability or coverage

FEEDBACK AND REVIEW PROCESS

Review Process Activities

Planning

- Defining the scope, which includes the purpose of the review, what documents or parts of documents to review, and the quality characteristics to be evaluated
- Estimating effort and timeframe
- Identifying review characteristics such as the review type with roles, activities, and checklists
- Selecting the people to participate in the review and allocating roles
- Defining the entry and exit criteria for more formal review types (e.g., inspections)
- Checking that entry criteria are met (for more formal review types)

Initiate review

- Distributing the work product (physically or by electronic means) and other material, such as issue log forms, checklists, and related work products
- Explaining the scope, objectives, process, roles, and work products to the participants
- Answering any questions that participants may have about the review

Individual review

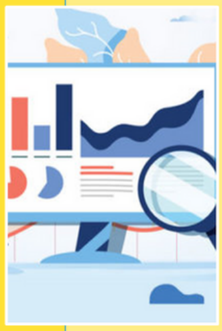
- Reviewing all or part of the work product
- Noting potential defects, recommendations, and questions

FEEDBACK AND REVIEW PROCESS

Review Process Activities

Issue communication & analysis

- Communicating identified potential defects
- Analyzing potential defects, assigning ownership and status to them
- Evaluating and documenting quality characteristics
- Evaluating the review findings against the exit criteria to make a review decision



Fixing & reporting

- Creating defect reports for those findings that require changes to a work product
- Fixing defects found (typically done by the author) in the work product reviewed
- Communicating defects to the appropriate person or team (when found in a work product related to the work product reviewed)
- Recording updated status of defects (in formal reviews), potentially including the agreement of the comment originator
- Gathering metrics (for more formal review types)
- Checking that exit criteria are met (for more formal review types)
- Accepting the work product when the exit criteria are reached

FEEDBACK AND REVIEW PROCESS

Roles and Responsibilities in Reviews

Author

- Creates the work product under review
- Fixes defects in the work product under review (if necessary)

Management

- Is responsible for review planning
- Decides on the execution of reviews
- Assigns staff, budget, and time
- Monitors ongoing cost-effectiveness
- Executes control decisions in the event of inadequate outcomes

Facilitator/ moderator

- Ensures effective running of review meetings (when held)
- Mediates, if necessary, between the various points of view
- Is often the person upon whom the success of the review depends

Review leader

- Takes overall responsibility for the review
- Decides who will be involved and organizes when and where it will take place

Reviewers

- May be subject matter experts, persons working on the project, stakeholders with an interest in the work product, and/or individuals with specific technical or business backgrounds
- Identify potential defects in the work product under review
- May represent different perspectives

Scribe (or recorder)

- Collates potential defects found during the individual review activity
- Records new potential defects, open points, and decisions from the review meeting (when held)

FEEDBACK AND REVIEW PROCESS

REVIEW TYPES

Informal review (e.g., buddy check, pairing, pair review)

- Main purpose: detecting potential defects
- Possible additional purposes: generating new ideas or solutions, quickly solving minor problems
- Not based on a formal (documented) process
- May not involve a review meeting
- May be performed by a colleague of the author (buddy check) or by more people
- Results may be documented
- Varies in usefulness depending on the reviewers
- Use of checklists is optional
- Very commonly used in Agile development

Walkthrough

- Main purposes: find defects, improve the software product, consider alternative implementations, evaluate conformance to standards and specifications
- Possible additional purposes: exchanging ideas about techniques or style variations, training of participants, achieving consensus
- Individual preparation before the review meeting is optional
- Review meeting is typically led by the author of the work product
- Scribe is mandatory
- Use of checklists is optional
- May take the form of scenarios, dry runs, or simulations
- Potential defect logs and review reports are produced
- May vary in practice from quite informal to very formal

Technical review

- Main purposes: gaining consensus, detecting potential defects
- Possible further purposes: evaluating quality and building confidence in the work product, generating new ideas, motivating and enabling authors to improve future work products, considering alternative implementations
- Reviewers should be technical peers of the author, and technical experts in the same or other disciplines
- Individual preparation before the review meeting is required
- Review meeting is optional, ideally led by a trained facilitator (typically not the author)
- Scribe is mandatory, ideally not the author
- Use of checklists is optional
- Potential defect logs and review reports are produced

Inspection

- Main purposes: detecting potential defects, evaluating quality and building confidence in the work product, preventing future similar defects through author learning and root cause analysis
- Possible further purposes: motivating and enabling authors to improve future work products and the software development process, achieving consensus
- Follows a defined process with formal documented outputs, based on rules and checklists
- Uses clearly defined roles, which are mandatory, and may include a dedicated reader
- Individual preparation before the review meeting is required
- Reviewers are either peers of the author or experts in other disciplines that are relevant to the work product
- Specified entry and exit criteria are used
- Scribe is mandatory
- Review meeting is led by a trained facilitator (not the author)
- Author cannot act as the review leader, reader, or scribe
- Potential defect logs and review report are produced
- Metrics are collected and used to improve the entire software development process, including the inspection process

FEEDBACK AND REVIEW PROCESS

SUCCESS FACTORS

- Defining clear objectives and measurable exit criteria. Evaluation of participants should never be an objective
- Choosing the appropriate review type to achieve the given objectives, and to suit the type of work product, the review participants, the project needs and context
- Conducting reviews on small chunks, so that reviewers do not lose concentration during an individual review and/or the review meeting (when held)
- Providing feedback from reviews to stakeholders and authors so they can improve the product and their activities
- Providing adequate time to participants to prepare for the review
- Support from management for the review process
- Making reviews part of the organization’s culture, to promote learning and process improvement
- Providing adequate training for all participants so they know how to fulfil their role
- Facilitating meetings