ISO/IEC/ IEEE 29119-1

TEST CONCEPTS

Ali Bahadori
Ali Hassani SokhtehSaraei

WHAT IS ISO/IEC/IEEE 29119 ABOUT?

IEEE 29119 ‘Software and systems engineering – Software testing’ is a collection of five software standards, developed by the Institute of Electrical and Electronics Engineers to provide a standardized approach to software testing.

- **IEEE 29119-1: Concepts and Definitions**
- **IEEE 29119-2: Test Processes**
- **IEEE 29119-3: Test Documentation**
- **IEEE 29119-4: Test Techniques**
- **IEEE 29119-5: Keyword-Driven Testing**

Let’s discover the 1st Part of IEEE 29119-1:2013 - Concepts and Definitions

NOTICE,

We would like to notify you that IEEE 29119-1:2013, which is the standard for software testing, has been updated to IEEE 29119-1:2021.

This update includes various changes and improvements that are geared towards enhancing the quality of software testing.

ISO/IEC/IEEE 29119-1:2013

It's called **'Concepts & Definitions'**, it defines the fundamental aspects, terminology, and methodology in software testing.

- **Section 1: Scope**
- **Section 2: Conformance**
- **Section 3: Normative references**
- **Section 4: Terms and Definitions**
- **Section 5: Software Testing Concepts**

We will focus on section 5 called **'Software testing Concepts'**.

ISO/IEC/IEEE 29119-1:2013

Some of the most important terms related to this document:

- Test Plan
- Test Design
- Test Case
- Test Procedure
- Test Script
- Test Result
- Dynamic Test
- Static Test
- Test Environment
- Test Data
- Test Automation
- Test Strategy
- Black-box Testing
- White-box Testing
- Gray-box Testing
- Verification
- Validation

& etc

WHY SOFTWARE TESTING IS NECESSARY?

the most important question

SOFTWARE TESTING IS NECESSARY BECAUSE:

- Information on the quality characteristics of the test item(s) is required by decision makers.
- The test item(s) being tested does not always do what it is expected to do.
- The test item(s) being tested needs to be verified.
- The test item(s) being tested needs to be validated.
- Evaluation of the test item(s) needs to be conducted throughout the software and system development life cycle.

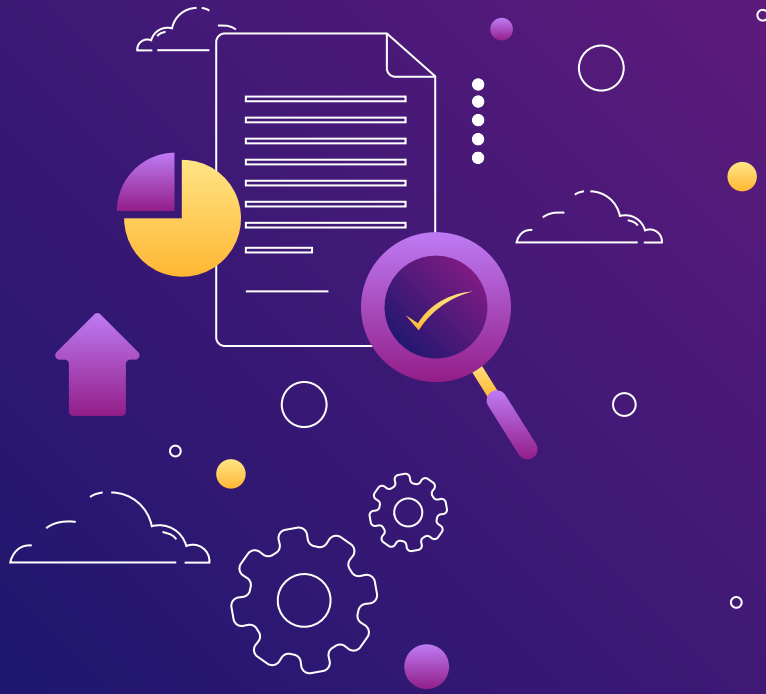
THE PRIMARY GOALS OF TESTING

- To provide information about the quality of the test item.
- Find defects in the test item prior to its release for use.
- Mitigate the risks to the stakeholders of poor product quality.

The provided information serves multiple purposes such as enhancing the quality of test items by eliminating defects, aiding management decisions by offering insight into quality and risk factors, and identifying organizational processes that permit defects to surface or go unnoticed so they can be modified.

INTRODUCTION TO SOFTWARE TESTING

The cluase 5-1 from 29119-1:2013



SOFTWARE TESTING

Software testing aims to **gather information about a software product and detect defects as early as possible within cost and schedule constraints**. Various considerations such as test planning, execution, automation, and analysis must be taken into account for effective testing.

SOFTWARE TESTING MISSION

Software testing should focus on providing information about a software product and finding as many defects as possible, as early as possible in the development process, under given constraints of cost and schedule.

TEST CONSIDERATIONS INCLUDE:

- Testing is a process. A process is a set of interrelated or interacting activities that transforms inputs into outputs.
- The organizational test process sets and maintains the test policies and test strategies that apply across the organization's projects and functions.
- Testing should be planned, monitored and controlled.
- Testing processes and sub-processes can be applied to any phase or level of testing or type of testing.

TEST CONSIDERATIONS INCLUDE:

- Testing entails examining a test item.
- Testing can be carried out on a product without executing the product on a computer.
- Static testing may also include the use of static analysis tools which find defects in the code or documents without the code executing.
- Dynamic testing consists of more than "just" running executable test items; it also includes both preparation activities and follow-up activities.

TEST CONSIDERATIONS INCLUDE:

- Verification is confirmation, through the provision of objective evidence, that specified requirements have been fulfilled in a given work item.
- Validation demonstrates that the work item can be used by the users for their specific tasks.
- Testing, whether static or dynamic, should aim to provide both types of confirmation, though with the expectation that confirmation will not be immediate because of the discovery of defects.

01 THE RULE OF TESTING IN VERIFICATION AND VALIDATION

It's important to note that this standard only addresses software testing, and does not cover other activities involved in verification and validation, such as V&V analysis or formal methods.

Therefore, in order to thoroughly verify and validate a product, an organization will need to use this standard in conjunction with other standards as part of a comprehensive engineering program. Annex A provides a hierarchy of verification and validation activities for reference.

02 EXHAUSTIVE TESTING

Due to the complexity of systems and software it is not possible to exhaustively test every single aspect of any given test item. Testers should recognize that exhaustive testing is not possible and that test activities should be targeted to best fulfill the test objectives for a test item. Risk-based testing is an approach that uses risk to direct test effort.

03 TESTING AS A HEURISTIC

Due to the complexity of systems and software it is not possible to exhaustively test every single aspect of any given test item. Testers should recognize that exhaustive testing is not possible and that test activities should be targeted to best fulfill the test objectives for a test item. Risk-based testing is an approach that uses risk to direct test effort.

SOFTWARE TESTING IN AN ORGANIZATION AND PROJECT CONTEXT

The cluase 5-2 from 29119-1:2013

Software testing is performed as a context-managed process. This means it should be planned, monitored and controlled. The context could be a development project or the on-going maintenance of an operational system.

The overall project plan should include consideration of the test activities to be performed as a part of the project. A Project Test Plan should reflect both the Organizational Test Policy and Organizational Test Strategy and deviations from these organizational guidelines.

A Project Test Plan includes a project test strategy and the project-specific decisions (including assumptions) used to derive this strategy.

The testing for a project will often be performed across a number of test sub-processes; each test sub-process may have a corresponding Test Plan consisting of a test sub-process strategy aligned with the project test strategy, and test sub-process specific detail.

The Project Test Plan describes the overall strategy for testing and the test processes to be used. It establishes the context of testing for the project by determining the objectives, practices, resources, and schedule; it also identifies the applicable test sub-processes.

The test plan also describes the appropriate test design techniques (static or dynamic) to use in order to complete the testing required by the particular sub-process plan.

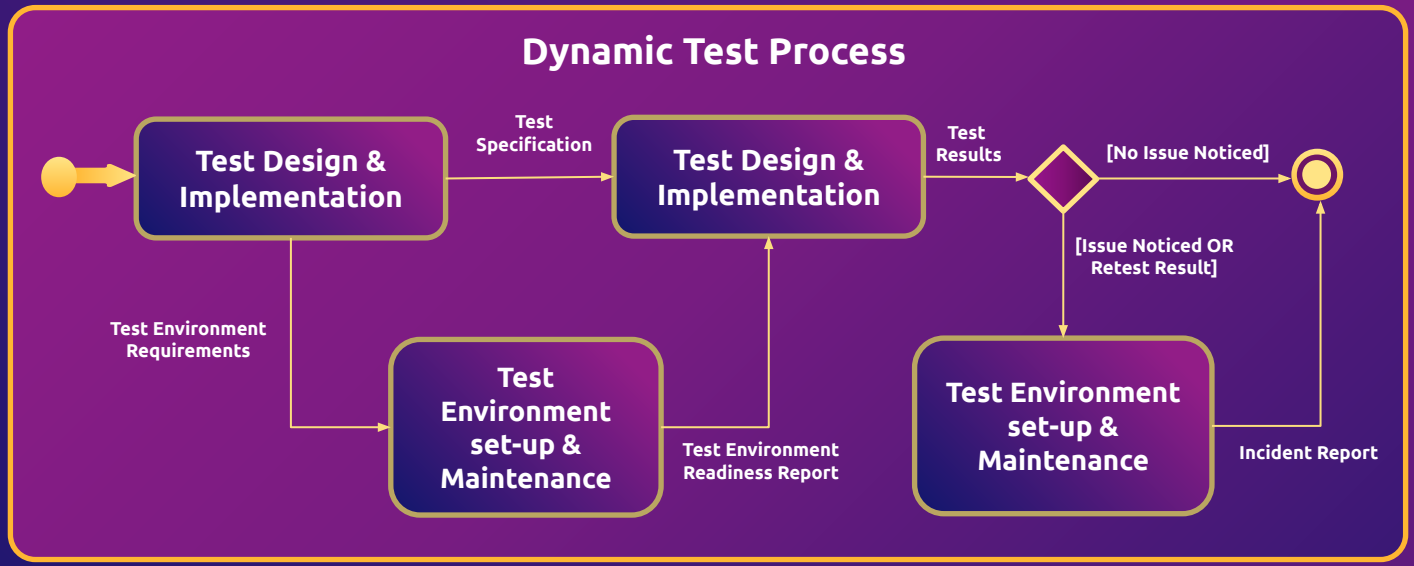
Test plans include a test strategy. Test strategies are tuned to the specific test project context.

01 THE TEST PROCESS

- Organizational Test Process
- The Management Processes
- Dynamic Test Processes

The Test Policy outlines software testing approach, guiding Organizational Test Strategy and test processes. Test Management manages per-project testing and produces reports. The Project Test Completion Report documents overall results for each project.

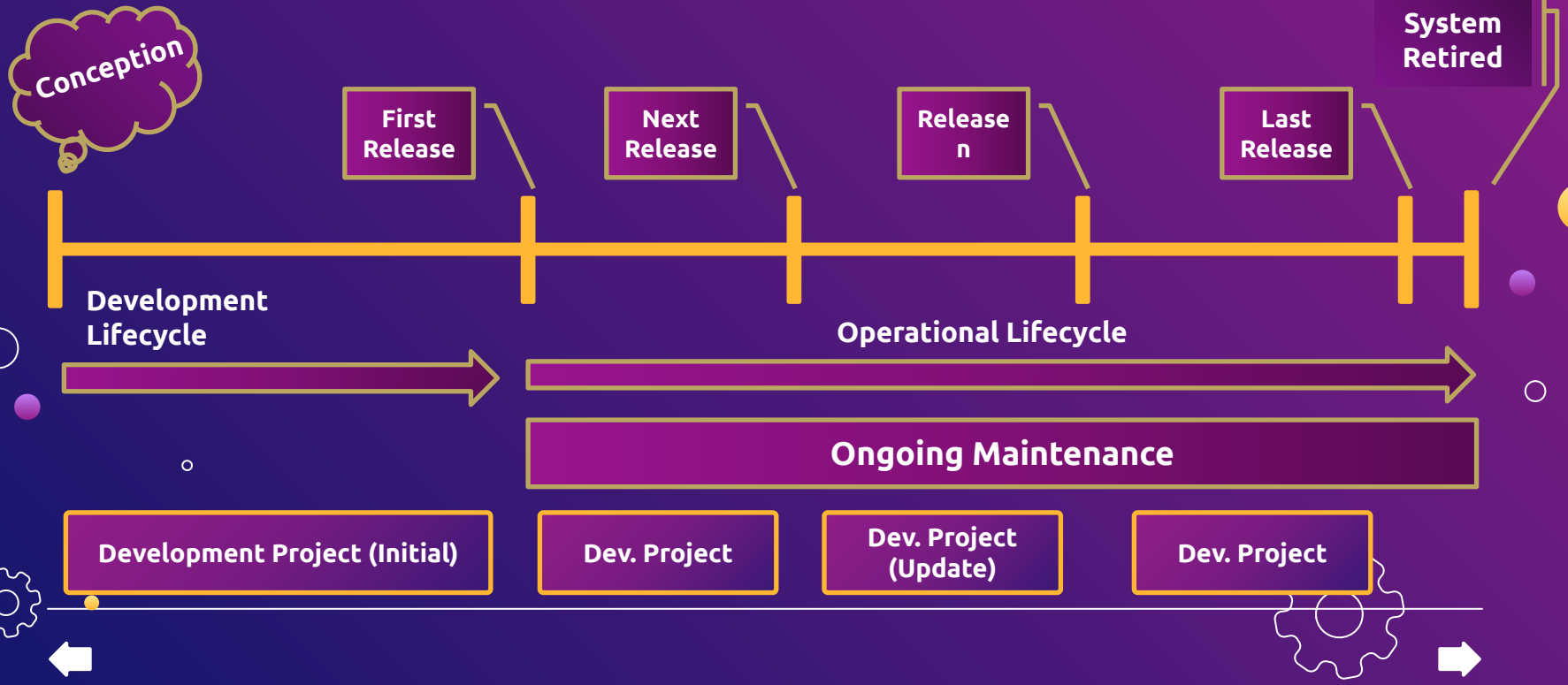
When testing a project, it's common to divide testing into smaller sub-processes like component testing, system testing, usability testing, and performance testing.



GENERIC TESTING PROCESSES IN THE SOFTWARE LIFE CYCLE

The cluase 5-3 from 29119-1:2013

Software has a life cycle from creation to retirement, and testing is part of its development and maintenance. ISO/IEC 12207 details software life cycles, while ISO/IEC 15288 outlines system life cycles.



Software life cycle has sub-cycles: development and operational. Dev is from start to release, while operation lasts until retirement. New versions treated as separate projects with testing. Maintenance keeps system working. Testing evaluates purchased software, framework in ISO/IEC 25051.

DEVELOPMENT PROJECT SUB-PROCESSES

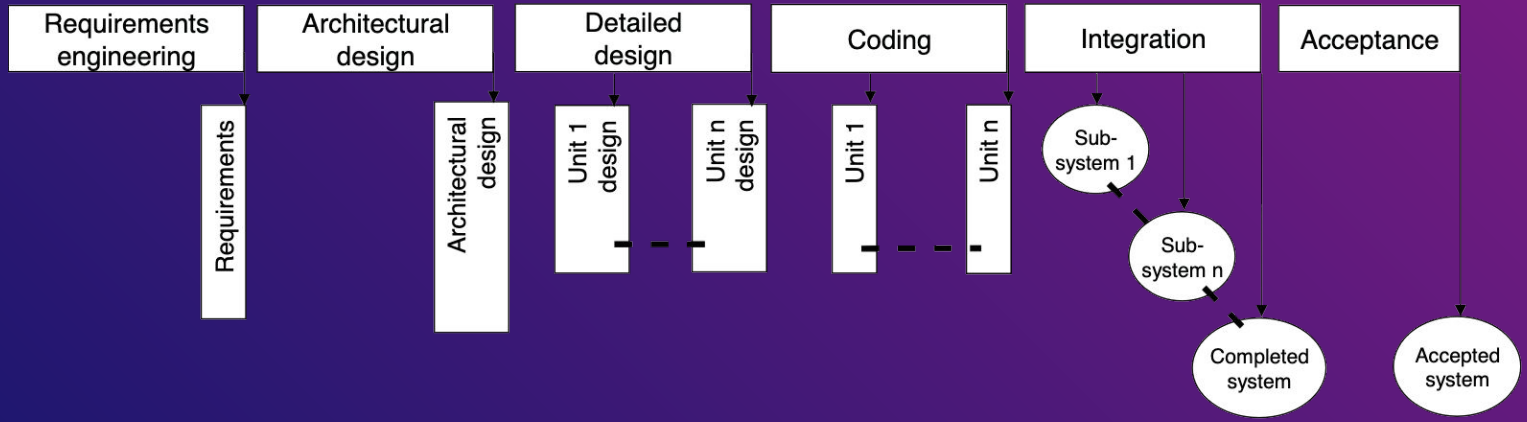
- Software and system development consists of common building blocks referred to as "development sub-processes"
- These include requirements engineering, architectural design, detailed design, and coding
- The approach to system development adopted will determine how these sub-processes are arrange

OUTPUT OF DEVELOPMENT SUB-PROCESSES

- In each development sub-process, something is produced, whether it is a highly-structured document or undocumented decisions
- The output may be formal or informal, depending on the process and methodology used
- Each sub-process may be performed once or repeated as necessary

DEVELOPMENT SUB-PROCESSES AND WORK PRODUCTS

Each work product, software system component, and complete software system is a potential test item



ON-GOING MAINTENANCE IN SOFTWARE LIFE CYCLE

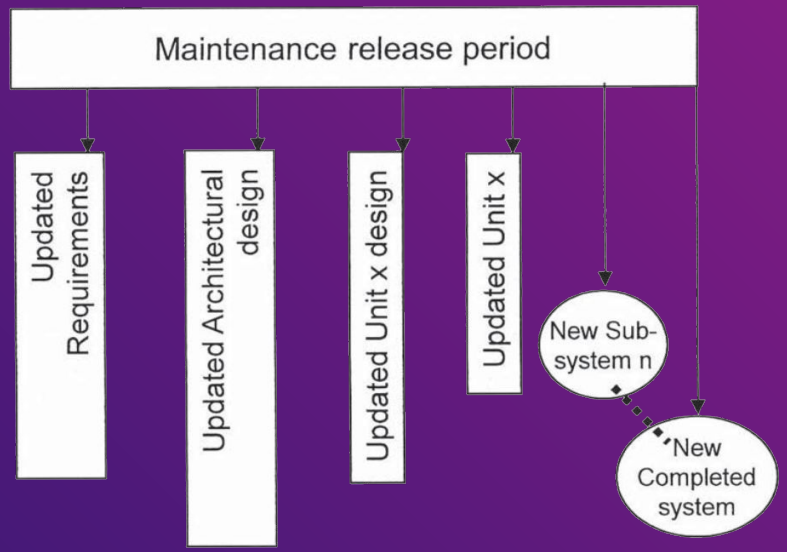
- On-going maintenance takes place during the operational part of the software life cycle, after initial development
- It may be managed in the context of an Application Management or IT Service Management process framework
- A project may be set up for continuous maintenance with a different financial model, often stated in a service level agreement (SLA)

OBJECTIVES OF ON-GOING MAINTENANCE

- The on-going maintenance process aims to maintain an acceptable or agreed level of reliability and availability of the system
- It involves producing new versions of the system with corrections of highest priority defects found in operation and introducing high priority changes to the functionality
- Versions may be released as the 'live' system on an ad hoc or fixed frequency basis

OUTPUTS OF MAINTENANCE RELEASE PERIOD

- The primary outputs of a maintenance release period include corrections, changes, and updates to work products such as requirements, design, code, and completed system
- Depending on the purpose of the release, each output may be produced in a single release period or only a subset is needed
- Maintenance release periods are repeated as necessary during the operational lifetime of the system



SUPPORT PROCESSES FOR SOFTWARE DEVELOPMENT LIFE CYCLE

Within an organization, support processes are required to aid the software development life cycle. Some of these are:

- Quality assurance; Project management
- Configuration management
- Process improvement.

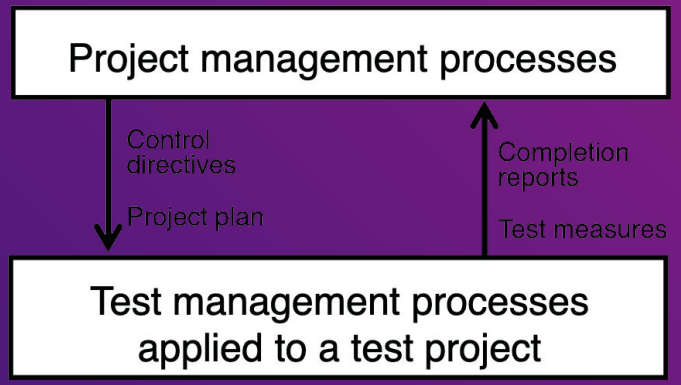
QUALITY ASSURANCE AND TESTING

Quality Assurance is a set of planned and systematic supporting processes and activities required to provide adequate confidence that a process or work product will fulfil established technical or quality requirements. This is achieved by the imposition of methods, standards, tools, and skills that are recognised as the appropriate practice for the context. The Quality Assurance process uses the test results and other information to investigate, rank and report any problem (including any risk) in the design, planning or execution of the software engineering processes.

Measures should be collected during testing, as they can provide information about the quality of test processes and/or test items and the effectiveness of their application on each project.

PROJECT MANAGEMENT AND TESTING

Project management is the process of planning and controlling the course of a project, which includes managing the test project within the overall project. The estimation, risk analysis, and scheduling of test activities should be incorporated into the overall project planning. Test measurements are analyzed by the test manager and communicated to the project manager, which may result in changes to the project plan. When a test sub-process or the test project is completed, a completion report is provided to the project manager.



CONFIGURATION MANAGEMENT AND TESTING

Configuration management ensures the integrity of work products. It includes processes such as unique identification, controlled storage, release audit, change control, and status reporting. Testing should verify if an organization's configuration management system meets its requirements. Work products from testing, such as test plans, specifications, and test environment configuration items can be placed under configuration management. All layers of the test process model interact with configuration management by providing and delivering configuration items.

PROCESS IMPROVEMENT AND TESTING

Process improvement involves changing an organization's processes to make them more efficient and effective in meeting business goals. The test processes and process improvement process interact in two ways: the test processes provide information for process improvement actions and can be subject to process improvement themselves. The process improvement process should obtain information from various sources to diagnose which test processes to improve and how to define the improved processes. The new processes should be monitored to assess whether they produce the expected return on investment.

RISK-BASED TESTING

The standard recognizes that exhaustive testing of a software system is impossible and outlines a variety of testing concepts to aid in choosing an appropriate sample for testing. The key premise of the standard is to perform optimal testing within given constraints and context using a risk-based approach. Risks can be categorized in different ways, and risk analysis is used to identify and score risks so that the perceived risks in the delivered system can be scored, prioritized, categorized, and subsequently mitigated. The risk profile is used to determine what testing should be performed on the project and to provide guidance to all the dynamic testing processes. Running test cases is a risk mitigation activity, and the process of test incident reporting is used to determine if the failed test case has resulted in an issue or if it requires further investigation.

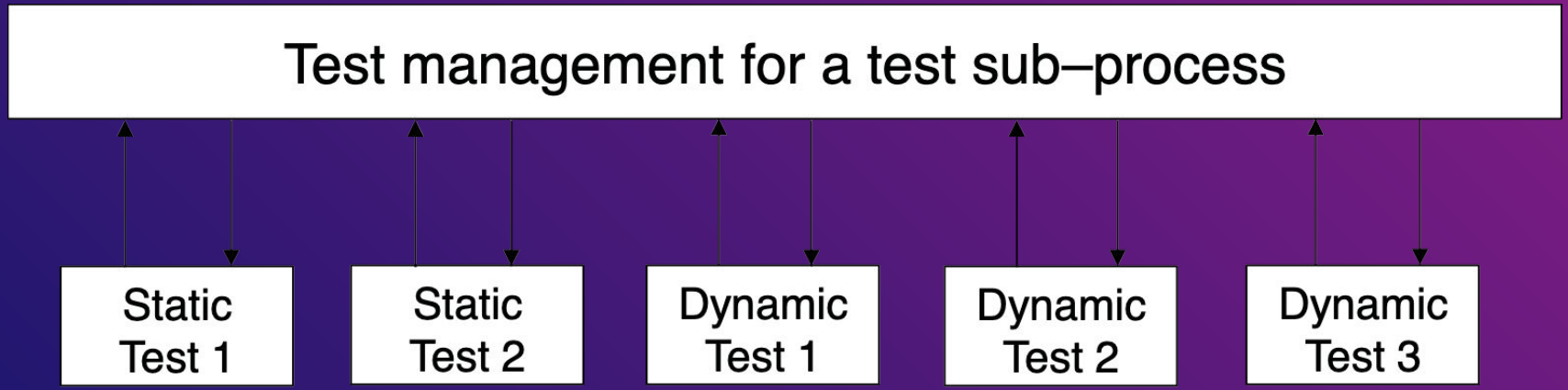
TEST SUB-PROCESS

- Test project structured as a number of test sub-processes based on the project test strategy. Each sub-process is managed by applying the test management processes to it. Includes both static and dynamic testing.
- Test Planning: Identifying test objective, scope, and risks associated with the sub-process. Guides strategy for the sub-process, including static and dynamic testing planned.
- Testing Scope: Specifying which features of the test item are to be tested and which are not, to ensure expectations concerning the scope of testing are clear.
- Retesting and Regression Testing: Repeating dynamic and static testing processes as needed to meet test completion criteria.

TEST SUB-PROCESS

- Test Objectives: Objectives include provision of information to risk management, assessment of product qualities, meeting stakeholder expectations, identifying defects and correct change implementation.
- Test Item: A test is performed on a test item against what is expected of the test item, examples of which include code-related items and document-related items.
- Quality Characteristics: Eight quality characteristics defined by ISO/IEC 25010 model for software quality, which include functional suitability, performance efficiency, compatibility, etc.

TEST SUB-PROCESS



TEST OBJECTIVES

A test is performed to achieve one or more objectives, which include provision of information to risk management, assessment of product qualities, meeting stakeholder expectations, identifying defects and correct change implementation.

TEST ITEM

A test is performed on a test item against what is expected of the test item, examples of which include code-related items and document-related items.

TESTING OF QUALITY CHARACTERISTICS

Testing measures the important quality characteristics for a given test item, defined by the eight quality characteristics outlined in the ISO/IEC 25010 model for software quality.

TEST BASIS

The "Test Basis" refers to the body of knowledge used to plan, design, implement, and manage software testing. This can include selecting test behavior, pass-fail criteria, inputs, environment, practices, and techniques. Examples of test basis may include documentation standards, customer requirements, expert knowledge, system architecture, and implementation details. Requirements can be classified into functional and non-functional categories, with functional requirements specifying what the item should do and non-functional requirements specifying how well it should behave. Non-functional requirements are related to the functionality and are typically associated with appropriate functional requirements.

RETESTING AND REGRESSION TESTING

Retesting is the process of evaluating whether a solution to an incident has fixed the original issue. This involves re-running the test case that produced the unexpected result and may require new test conditions and cases to be identified and written. Regression testing is selective testing of a system or component that has been previously tested to ensure that modifications have not caused unintended side-effects and that the system still meets its original requirements. It is important to consider both retesting and regression testing when planning testing and allocate sufficient time in the test execution schedule for both activities.

TEST DESIGN TECHNIQUES

Test design techniques can assist testers in finding defects effectively and efficiently. There are two types of test design techniques: static and dynamic. Static testing identifies apparent defects or anomalies in documentation or source code, while dynamic testing forces failures in executable test items.

Static testing includes activities like static code analysis, cross-document traceability analysis, and reviews to find issues and provide information about software products. Static test design techniques aim to find defects, but they can also serve other purposes like knowledge exchange, gaining consensus, and preventing future defects. The type of static test design technique to use depends on the risks involved and the secondary purpose.

TEST DESIGN TECHNIQUES

Dynamic testing uses techniques to identify test conditions, coverage items, and subsequently test cases to be executed on a test item. There are three main categories of dynamic test design techniques: specification-based, structure-based, and experience-based. Specification-based techniques derive test cases from a test basis describing the expected behavior of the test item. Structure-based techniques derive test cases from a structural feature, such as the structure of source code or a menu structure. The choice of which technique to use depends on the nature of the test basis and the risks involved. Examples of dynamic test design techniques covered in ISO/IEC/IEEE 29119-4 include Boundary Value Analysis, State Transition Testing, and Branch Testing.

TEST PRACTICES

This section introduces various practices for planning and implementing testing for a project and outlines some of the options available during test planning.

The risk-based approach to testing, as described in clause 5.4, has been widely adopted and is the fundamental approach for the ISO/IEC/IEEE 29119 set of standards. This section explains that different practices can be used for planning and implementing testing and that the choice of test strategy is determined by various risks.

REQUIREMENTS-BASED TESTING

The main purpose of requirements-based testing is to ensure that the requirements of the test item have been addressed during testing and to determine whether the test item meets end-user requirements. This section explains that requirements-based testing can be supported by other test practices and that prioritization can be used to test higher-risk requirements more thoroughly and earlier.

MODEL-BASED TESTING

Model-Based testing uses a fundamentally different practice, but still based on a model of the expected behavior. The model has to be formal enough and/or detailed enough so that useful test information can be derived from it. This section also explains that the advantages of using model-based testing include the generation of test information, improved levels of automation, and early identification of some kinds of errors.

MATHEMATICAL-BASED TESTING

This type of testing uses mathematical techniques to plan and design tests, reduce human bias, and objectively sample the test case space. Combinatorial testing and random test case selection are two techniques commonly used, and statistical modeling can also be employed. Automated tools are typically required due to the large number of inputs generated.

EXPERIENCE-BASED TESTING

This approach draws on previous testing experience, knowledge of software and systems, domain knowledge, and metrics from previous projects. Error guessing is a specific technique described in ISO/IEC/IEEE 29119-4, and other practices include exploratory testing, software attacks, and ad hoc testing.

SCRIPTED AND UNSCRIPTED TESTING

These are two approaches to testing that are not mutually exclusive and are often combined in a hybrid practice. Scripted testing involves pre-designed and documented test procedures, while unscripted testing relies on intuition, curiosity, and previous test results to guide testing. The risk profile of the test item is the primary consideration in deciding which approach or hybrid to use.

- Scripted Testing: Repeatable tests, good auditability, reusable artifacts, less adaptable to system changes, less stimulating for testers.

- Unscripted Testing: Ability to follow real-time ideas, tailor tests to system behavior, quick exploration of the test item, not repeatable, requires experienced testers, provides little record of test execution.

AUTOMATION IN TESTING

Test automation involves automating many of the tasks and activities described in the Test Management and Dynamic Testing processes. Test automation tools can be used for test case management, test monitoring and control, test data generation, static analysis, test case execution, test environment implementation and maintenance, and session-based testing.

DEFECT MANAGEMENT

Defect management involves investigating and documenting test incident reports and retesting defects when required. It is not directly covered by the ISO/IEC/IEEE 29119 set of standards, but concepts and processes can be found in other standards such as ISO/IEC 12207, ISO/IEC 20000, and IEEE 1044.