

<epam>

The History of Software Testing

Software Testing Introduction



TRAINING
C E N T E R

— <epam> —

20th Century: 50th-60th

Mathematical
Approach

Exhaustive
Testing

Debugging

This is not the “Testing”, this is just some sort of activity during
the Development

50th-60th

70th

80th

90th

2000th

Now



20th Century: 70th

Prove that software works
(Positive Testing)

Prove that software fails
(Negative Testing)

Both first attempts failed, but the basis for modern approaches
has been built



20th Century: 80th

Lifecycle
Integration

Error Prevention
Approach

First Automation
Attempts

Software Testing is recognized globally. A lot of methodologies, approaches, standards, etc. appears.

50th-60th

70th

80th

90th

2000th

Now

20th Century: 90th

Quality
Assurance

Integration with
Development

Numerous Tools
and Systems

For the first time Software Testing looks a lot like today

50th-60th

70th

80th

90th

2000th

Now



21st Century Beginning

Agile Testing

TDD and other
approaches

Automation
Everywhere

Software Testing becomes inseparable part of software
development process

50th-60th

70th

80th

90th

2000th

Now



Modern Situation

Agile Testing

Testing as a part
of...

Testing as a basis
for...

Software Testing becomes a part of Development, DevOps and other activities. It becomes a basis for CI/CD/etc. approaches.

50th-60th

70th

80th

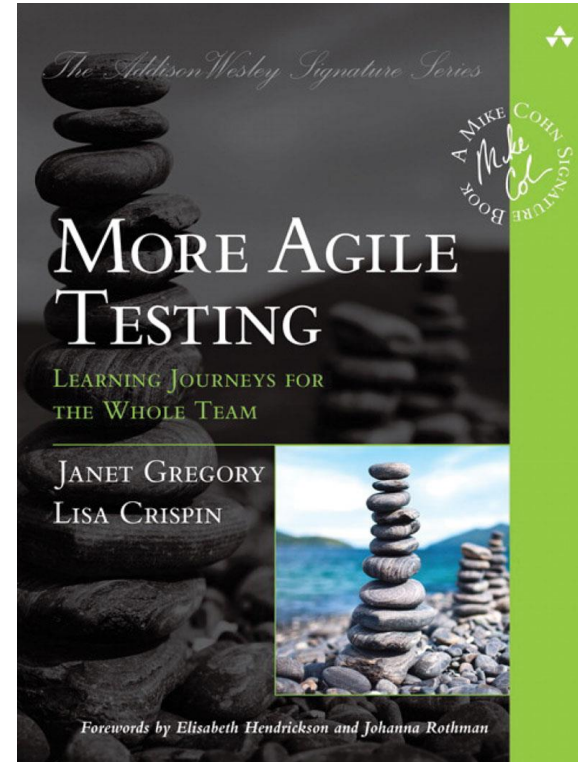
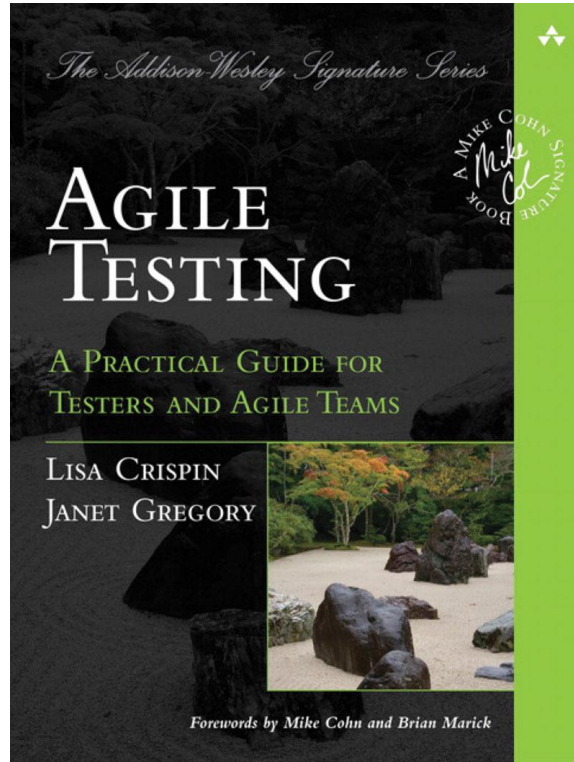
90th

2000th

Now



More About Our Days



<epam>

The History of Software Testing

Software Testing Introduction



TRAINING
CENTER

— <epam> —



Why Software Testing is Important and What to Test

Software Testing Introduction



TRAINING
CENTER



For Individual Users (and B2C Businesses)

Years ago

Now

You want a Tool? Go write it
yourself!

You want a Tool? Please,
select from the following
100500 options.

For Enterprise (and B2B Businesses)

Individual Users

Do like quality, but often fall victims of aggressive marketing.

Enterprise Customers

They count. And count. And count again. And again. The quality affects this calculations essentially.

For Everyone

Trains, airplanes,
ships, etc...

Nuclear power
plants...

Medical
equipment...

Today software is literally **EVERYWHERE**. Smartphone failure is bad, but airplane crash is incomparably worse.

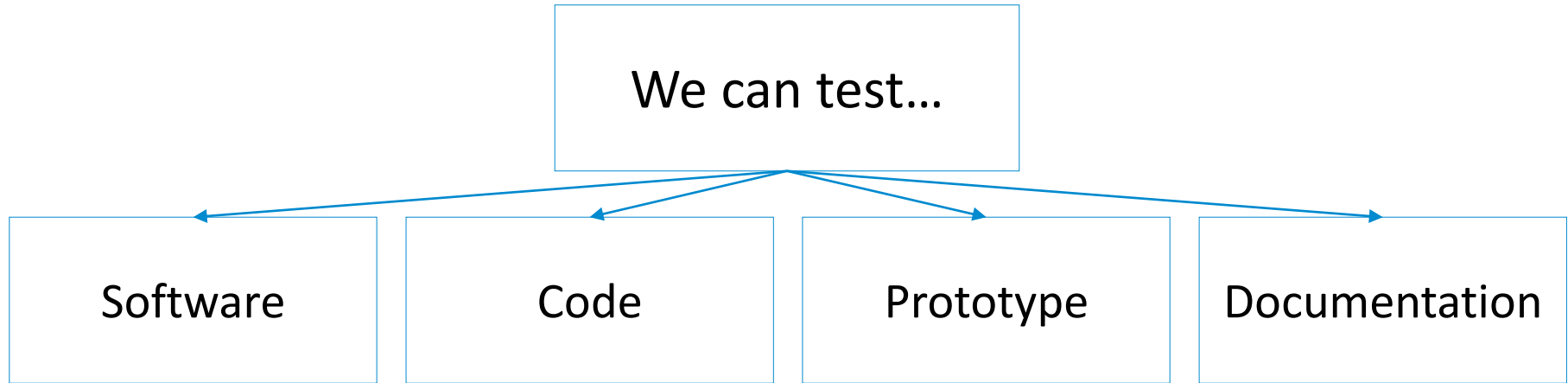
What to Test: Main Testing Areas

We can test...



Literally EVERYTHING

What to Test: Main Testing Areas





Why Software Testing is Important and What to Test

Software Testing Introduction



TRAINING
CENTER





Soft Skills and Hard Skills of Software Tester

Software Testing Introduction



TRAINING
CENTER



Enhanced responsibility

Good communication skills

Presentation skills

Performance and goal oriented approach

Observation and attentiveness to details

Flexible thinking and ability to learn fast

Good abstract and analytical thinking

Tendency to experimenting

Hard Skills

Good English (B2+)

Programming (including web and mobile development) basics

Databases (and SQL) basics

OS administration basics

Network administration basics

Virtualization and cloud computing basics

Automated testing basics

General IT knowledge

Hard Skills

Good English

General IT knowledge

This is the ultimate minimum to start with!



Soft Skills and Hard Skills of Software Tester

Software Testing Introduction



TRAINING
CENTER





Basic Software Testing Terminology

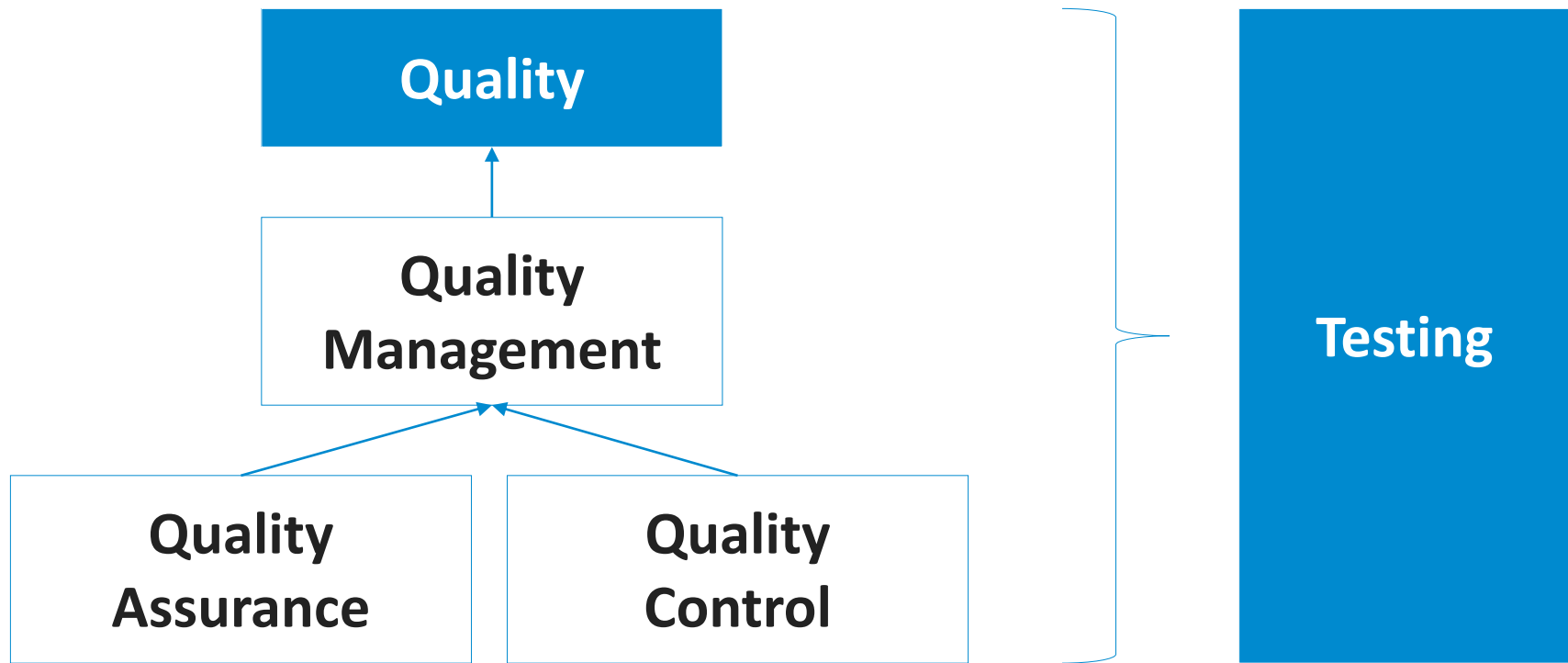
Software Testing Introduction



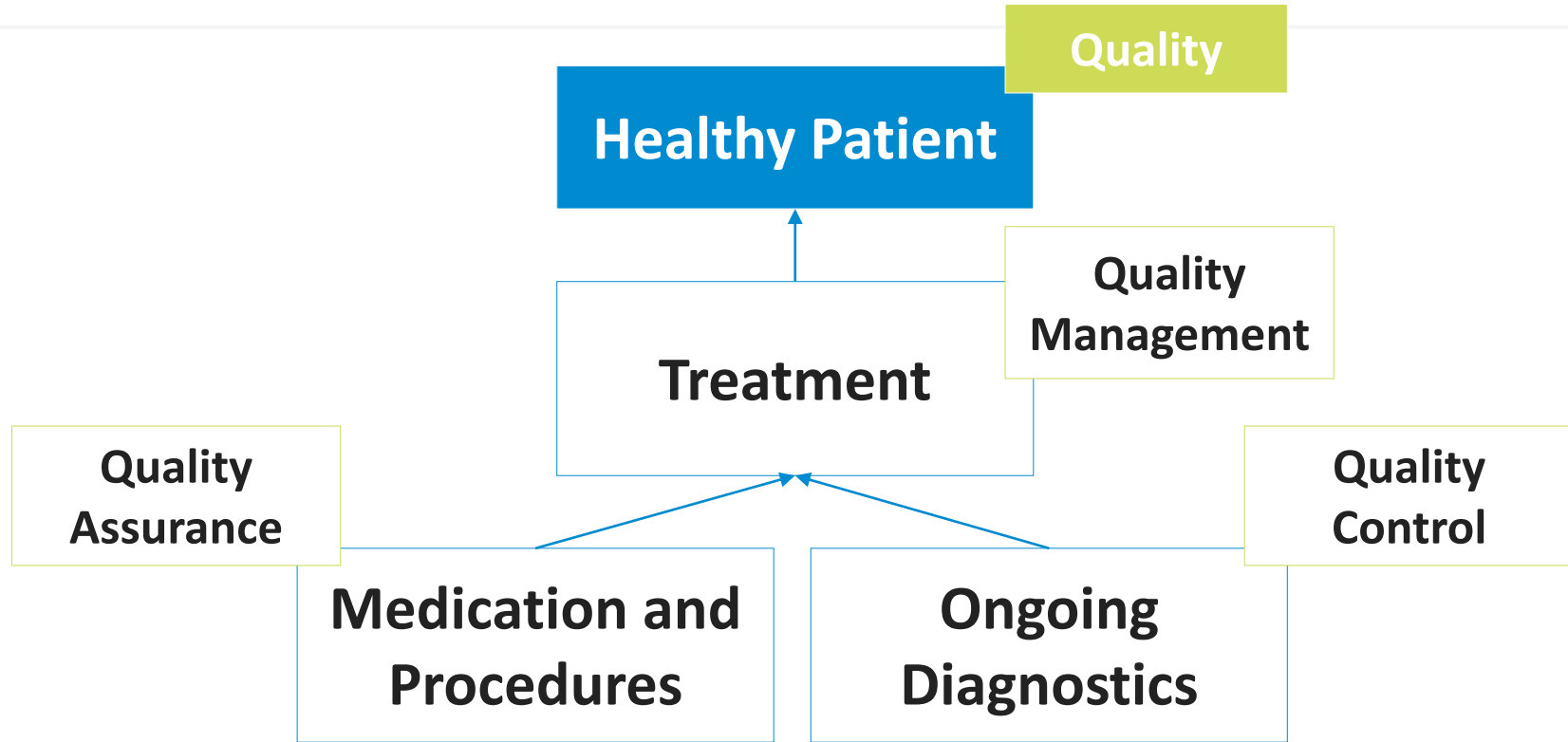
TRAINING
CENTER



Testing and Quality



Testing and Quality by Example



Testing – the process concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.

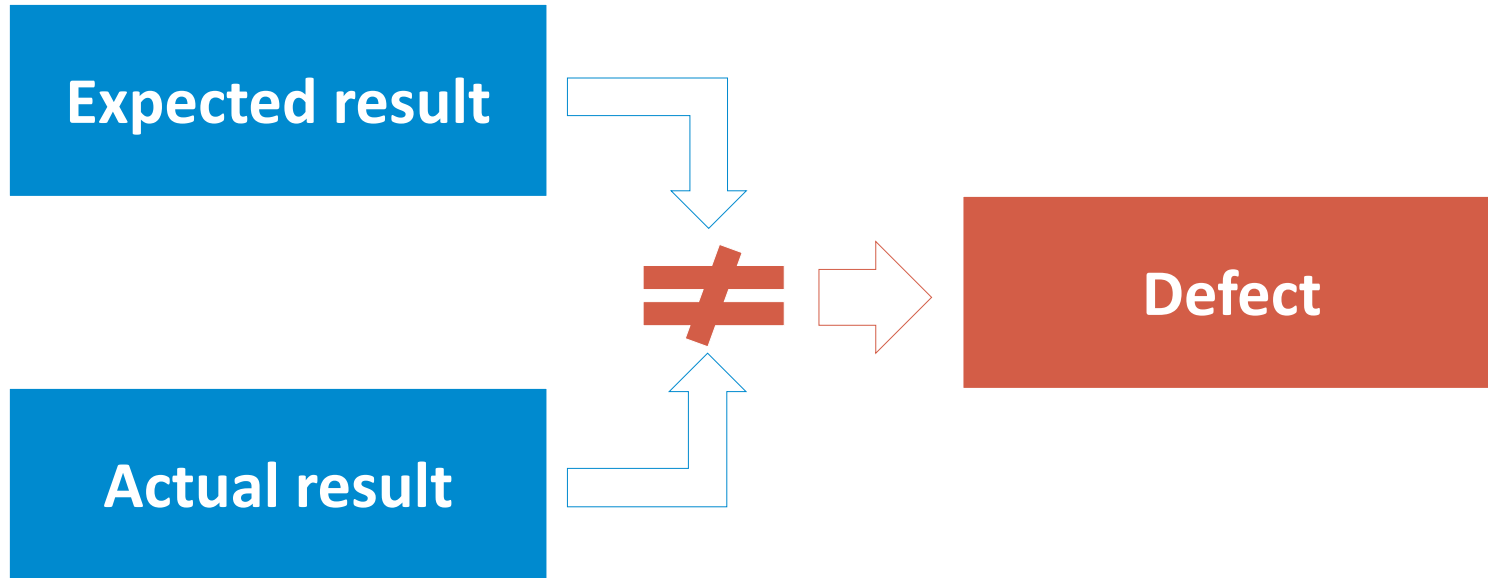
Quality – the degree to which a component, system or process meets specified requirements and/or user/customer needs and expectations.

Quality management – coordinated activities to direct and control an organization with regard to quality.

Quality assurance – part of quality management focused on providing confidence that quality requirements will be fulfilled.

Quality control – the operational techniques and activities, part of quality management, that are focused on fulfilling quality requirements.

Defect, Expected Result, Actual Result



Defect – an imperfection or deficiency in a work product where it does not meet its requirements or specifications.

Expected result – the predicted observable behavior of a component or system executing under specified conditions, based on its specification or another source.

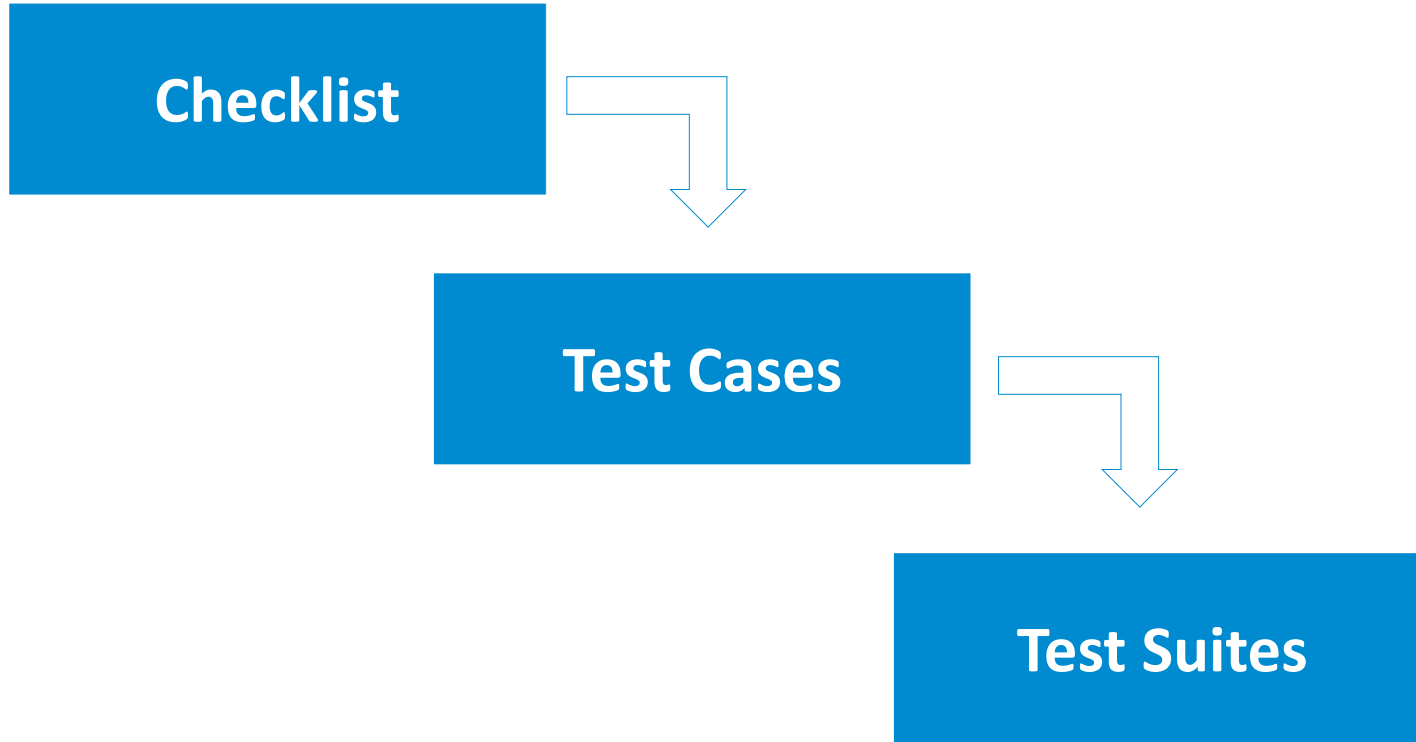
Actual result – the behavior produced/observed when a component or system is tested.

Defect Report: Sample

ID	Summary	Descriptions	Steps to reproduce
2352	Multiple instances crash on competition for the SOURCE_DIR	With 2+ instances of the application running with the same input directory configured, they compete for files in this directory that leads to multiple file operation failures and (finally) to crash of some instances. Exp: all instances work (not crashes); if a file operation fails, the affected instance makes a log record and retries the operation. Act: the affected instance crashes. Req: UR-12.	<ol style="list-style-type: none">1. Start several (3-5-7+) instances of the app with the same valid parameters.2. Start putting valid files into the SOURCE_DIR. Bug: one by one most instances crash.

Reproducibility	Severity	Priority	Symptom	Workaround	Comments	Attachments
Always	Critical	Normal	System crash	No	It looks like some semaphore or synchronization mechanism is either broken or do not take the SOURCE_DIR into account. Or some file operation failure handler does not work as it has to.	-

Checklist, Test Case, Test Suite



Checklist – a set of ideas.

Test case – a set of preconditions, inputs, actions (where applicable), expected results and postconditions, developed based on test conditions.

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

Checklist: Sample

- Configuration and start:
 - With correct parameters:
 - Values for SOURCE_DIR, DESTINATION_DIR, LOG_FILE_NAME are passed to the application. These values (all of them) should contain spaces and Cyrillic symbols. Repeat this test for Windows and *nix file systems (pay attention to directory separators ("/" и "\") and logical drives).
 - LOG_FILE_NAME value is omitted.
 - Without any parameter.
 - With some mandatory parameters omitted.
 - With incorrect parameters:
 - Wrong path for SOURCE_DIR.
 - Wrong path for DESTINATION_DIR.
 - Wrong file name for LOG_FILE_NAME.
 - DESTINATION_DIR is inside SOURCE_DIR.
 - DESTINATION_DIR and SOURCE_DIR are the same.

Test Case: Sample

Id	Priority	Requirement	Module	Submodule	Steps	Expected Results
CPT.17	B	-	Scanner	Traverser	<p>Multiple copies of the app working simultaneously, file operations conflict</p> <p>Preparations:</p> <ul style="list-style-type: none">Create three separate directories in the root folder of any logical drive (or in the user home directory for *nix systems): one for INPUT files, one for OUTPUT files, one for LOG file.Create several files of the maximum supported size, supported formats, and supported encodings. <ol style="list-style-type: none">Start the first copy of the app with corresponding paths from preparations as parameters (use any log file name).Start the second copy of the app with the same parameters (see step 1).Start the third copy of the app with the same parameters (see step 1).Change the process priority for the second copy to "high" and for the third copy to "low".Copy the prepared set of files (see preparations section) into the INPUT directory.	<ol style="list-style-type: none">All three copies of the app are started, the log file contains three sequential startup records.Files gradually move from INPUT to OUTPUT directory. The console and the log file contains messages about successful file conversion for each file. The console and the log file may also contain the following error messages:<ol style="list-style-type: none">"source file inaccessible, retrying"."destination file inaccessible, retrying"."log file inaccessible, retrying". <p>The key success indicator for this test is the successful conversion of all files while no copy of the app should crash. The log file should contain at least one record (three – maximum) for each converted file.</p> <p>Mentioned above error messages are OK too. They may appear, or may not – it depends on too many conditions and too hard to predict.</p>

Test Suite: Sample


CPT.4	B	DS-5.2	File processor	Encoding converter	Conversion of file with maximum allowed size
CPT.5	B	DS-5.2	Scanner	Traverser	Too big files conversion
CPT.6	B	-	Scanner	Traverser	Empty files conversion
CPT.7	C	-	Scanner	Traverser	Directory tree inside the input directory
CPT.8	C	-	File processor	Encoding converter	Multiple encodings inside one file
CPT.9	B	DS-2.4	Scanner	Traverser	No access rights to the input directory
ST.1	A	DS-2.2	App loader	Error handler	Start with no parameters
ST.2	A	DS-2.2	App loader	Error handler	Start with some mandatory parameter omitted
ST.3	B	DS-2.4	App loader	Error handler	Invalid path to the log file
ST.4	A	DS-2.4	App loader	Error handler	Invalid paths to the input/output directory
ST.5	A	DS-2.4	App loader	Error handler	Incorrect start parameters, nonexistent paths
ST.6	B	-	App loader	Parameters analyzer	Valid parameters with spaces and Cyrillic symbols
ST.7	A	DS-5.1	File processor	Encoding converter	Conversion from all supported encodings

Now we can compose the following (or any other) sets:

- If we want tests for “Error handler” submodule, we get the following test cases: CPT.2, CPT.3, ST.1, ST.2, ST.3, ST.4, ST.5.
- If we want tests with the priority “A”, we get the following test cases: ST.1, ST.2, ST.4, ST.5, ST.7.
- If we want tests with the priority “B” for the “App loader” module, we get the following test cases: CPT.1, CPT.2, CPT.3, ST.3, ST.6.
- If we want tests for the “BR” section of Requirements, we get the following test cases: CPT.2, CPT.3.
- If we want tests with the priority “C” for the “File processor” module and the “DS” section of Requirements, we get the following test case: CPT.14.
- And so on: we choose the set of criteria and get the resulting set of test cases.

Please, visit this URL

<https://glossary.istqb.org>

**ISTQB Glossary**[Advanced Options ▼](#)[Reports](#)[English](#)

Click on a term to see more information about a term

abuse case	A use case in which some actors with malicious intent are causing harm to the system or to other actors.
acceptance criteria	The criteria that a component or system must satisfy in order to be accepted by a user, customer, or other authorized entity.
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.
accessibility	The degree to which a component or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use.
accessibility testing	Testing to determine the ease by which users with disabilities can use a component or system.



Basic Software Testing Terminology

Software Testing Introduction



TRAINING
CENTER





Software Development Models and Testing

Software Testing Introduction

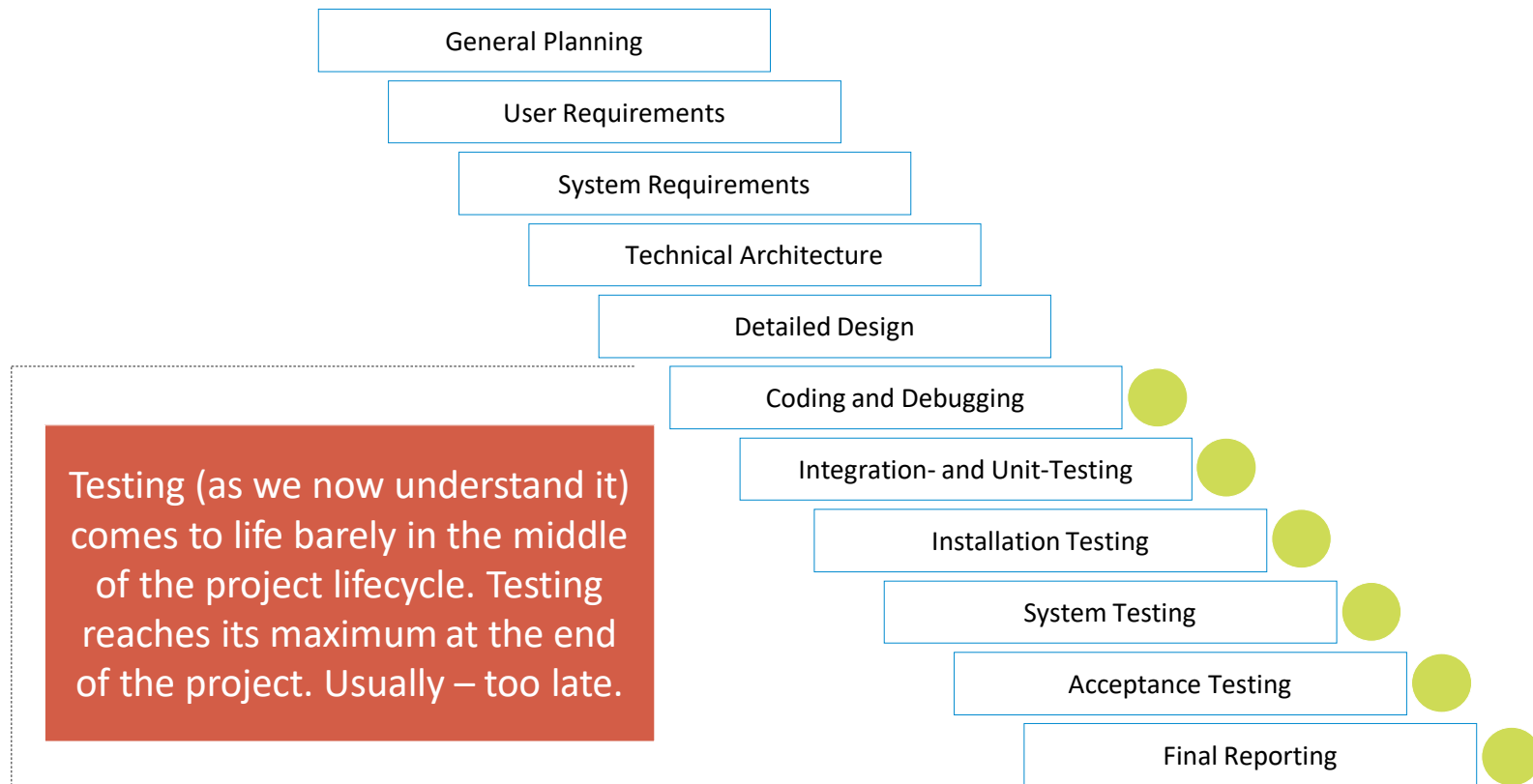


TRAINING
CENTER

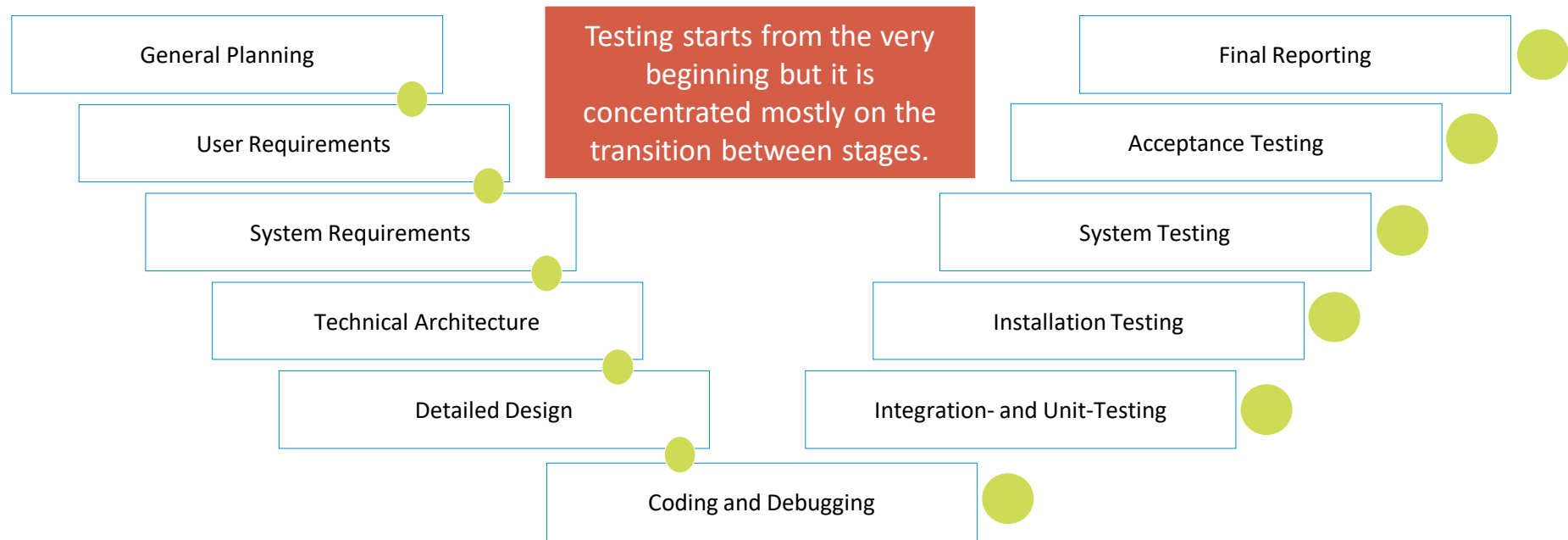


This is **NOT** a detailed review or a comparison of Software Development Models. This is just a quick answer to “Where is the Testing in the ... Model?”

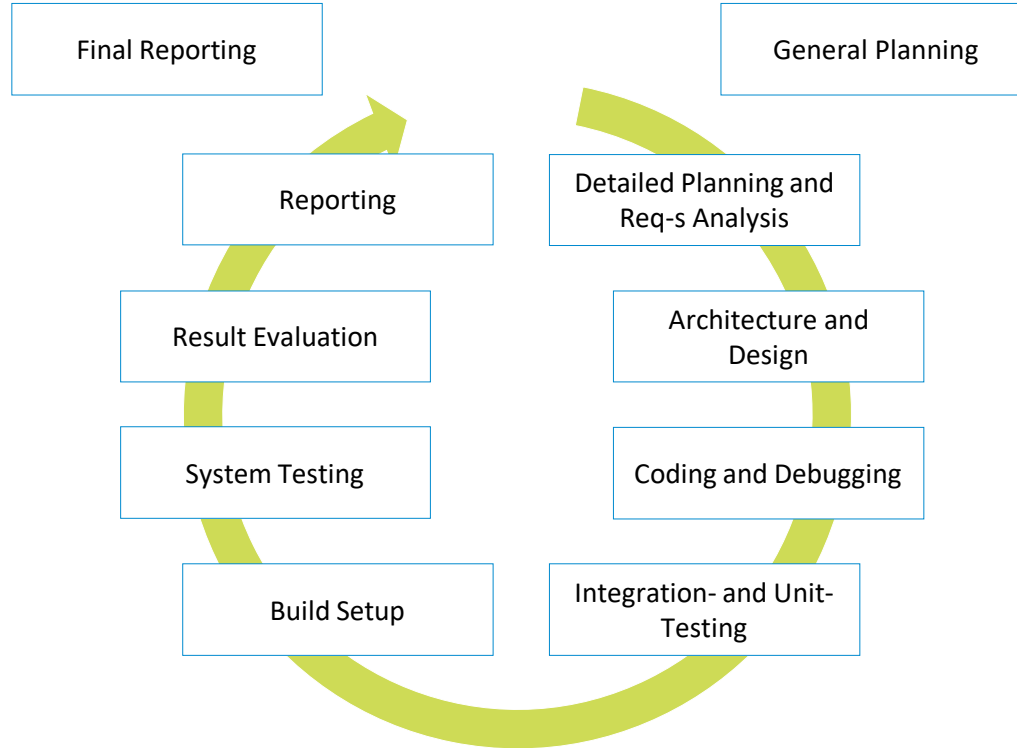
Waterfall



V-Model

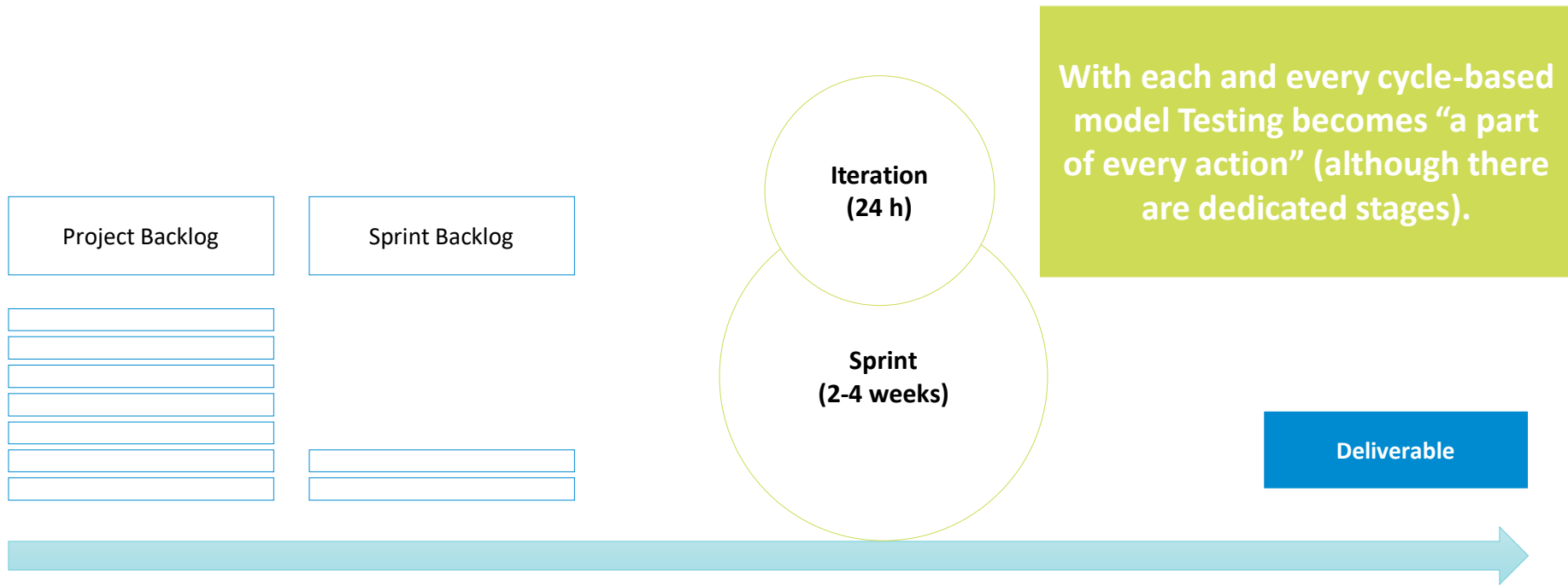


Iterative Incremental Model



With each and every cycle-based model Testing becomes “a part of every action” (although there are dedicated stages).

Agile Model



And what about...

RAD Model

Spiral Model

Prototype Model

Any Other Model

With non-iterative models Testing lies within dedicated stages and between stages.

With iterative models Testing not only has dedicated stages, but fills any other activities like air or water.

With such approach Testing helps us to find defects ASAP or even to prevent them.



Software Development Models and Testing

Software Testing Introduction



TRAINING
CENTER





Software Testing Lifecycle

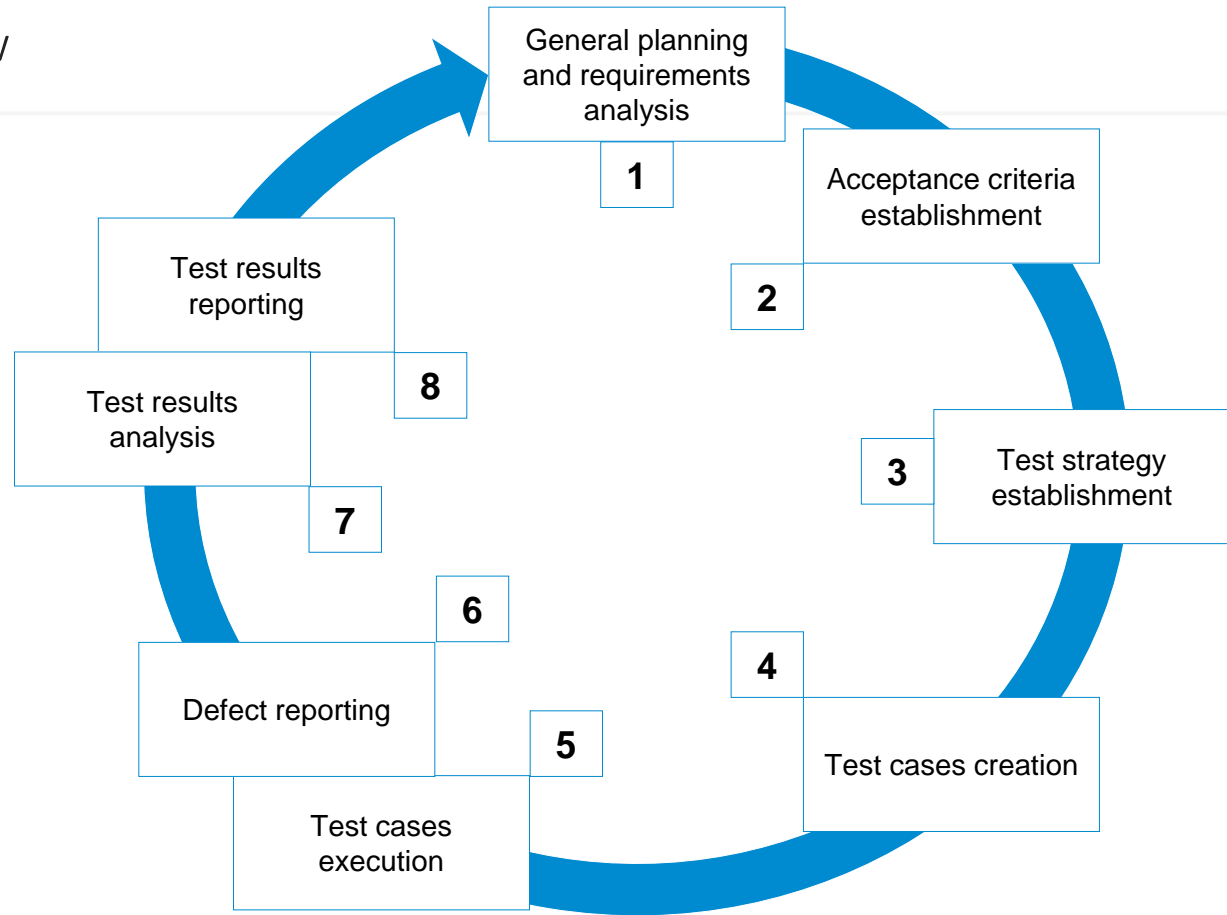
Software Testing Introduction



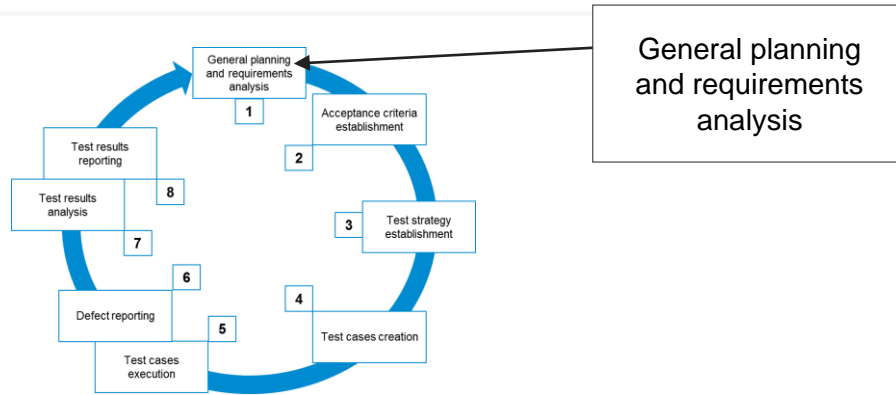
TRAINING
CENTER



General Overview



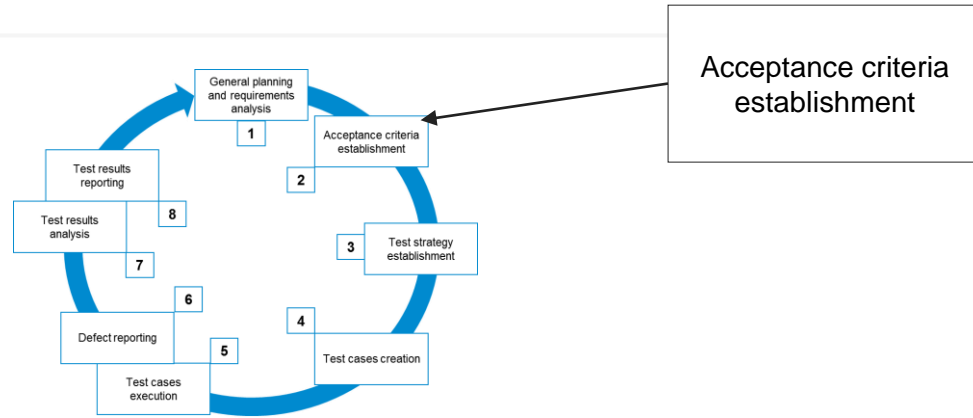
General planning and requirements analysis



1. General planning and requirements analysis

Here we have to find out: what to test; how much work is ahead; what difficulties we may face; do we have all necessary resources; are requirements good enough.

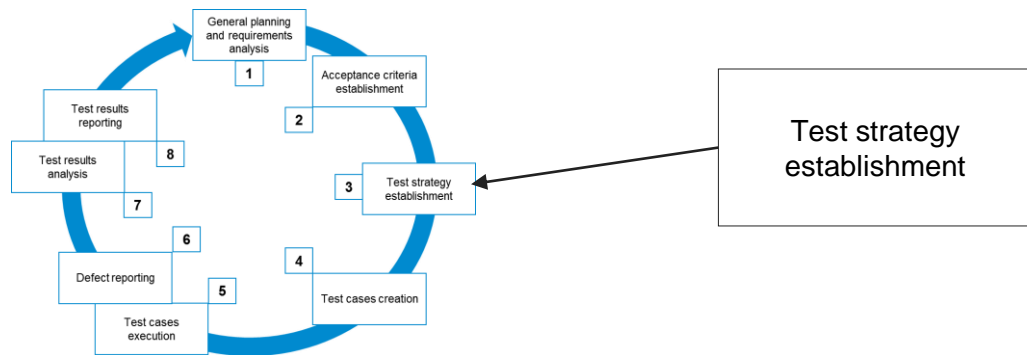
Acceptance criteria establishment



2. Acceptance criteria establishment

Here we have to establish or adjust metrics and criteria for test process to start, pause, resume, complete or abort. We also have to know key quality criteria and goals for the current test cycle.

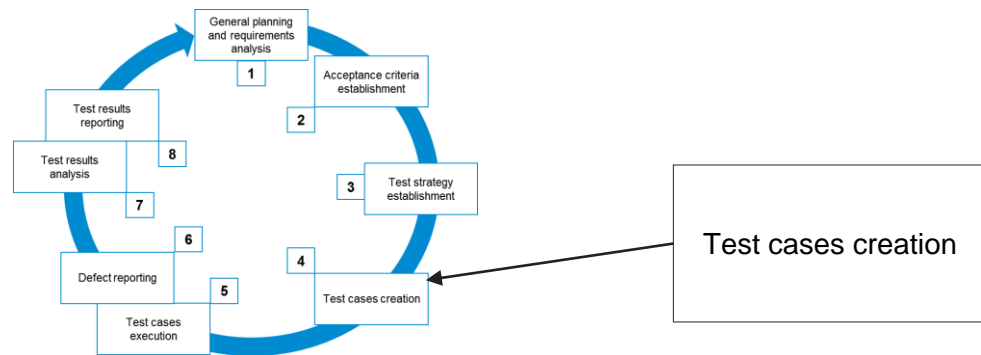
Test strategy establishment



3. Test strategy establishment

Once again we return to planning to find out HOW shall we achieve all those goals and criteria from the previous step. Here we speak about approaches, tools, schedule, roles, responsibility and so on.

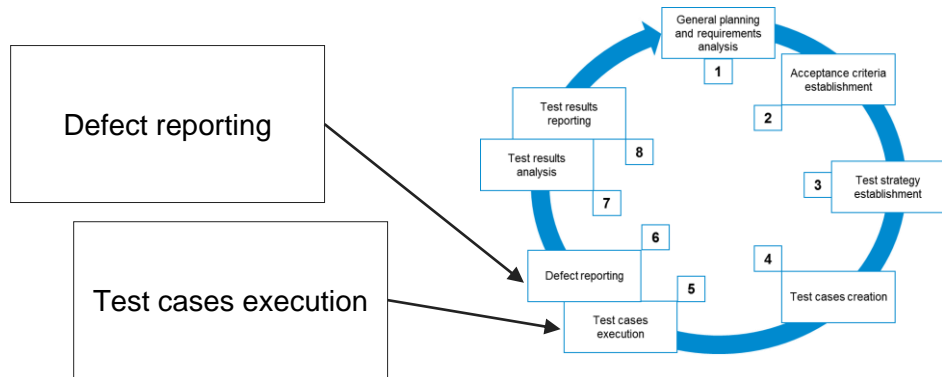
Test cases creation



4. Test cases creation

Here we create, review, adjust, rework (and so on) checklists, test cases, test scenarios and other similar artifacts.

Test cases execution and defect reporting

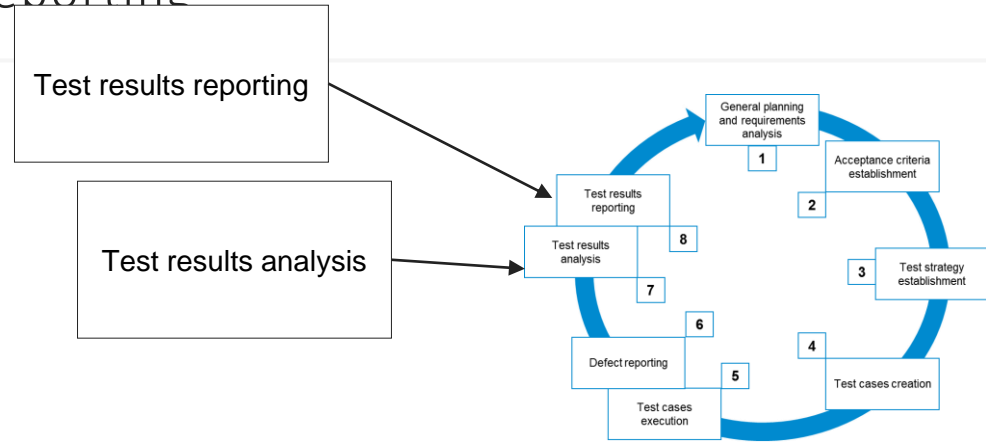


5. Test cases execution

6. Defect reporting

These two stages are inseparable as we report defects during test cases execution and defects detection.

Test results analysis and reporting



7. Test results analysis

8. Test results reporting

Here we analyze if we managed to achieve the goals set during stages 1-3. The result of such analysis is presented in Test Result Report and used as the basis for the planning of the next test cycle.



Software Testing Lifecycle

Software Testing Introduction



TRAINING
CENTER





Software Testing Classification

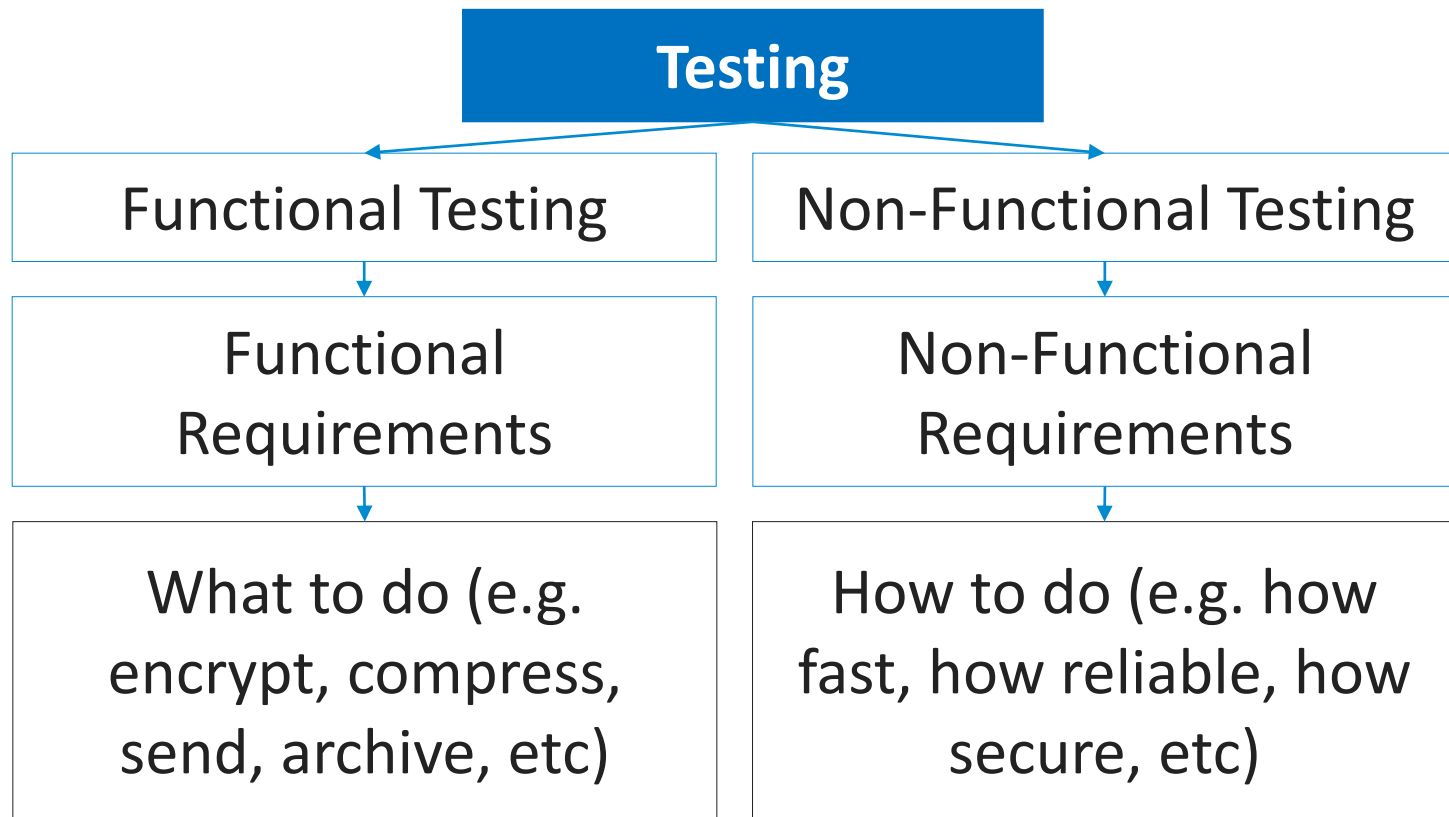
Software Testing Introduction



TRAINING
CENTER



Two Fundamental Approaches: Functional and Non-Functional Testing



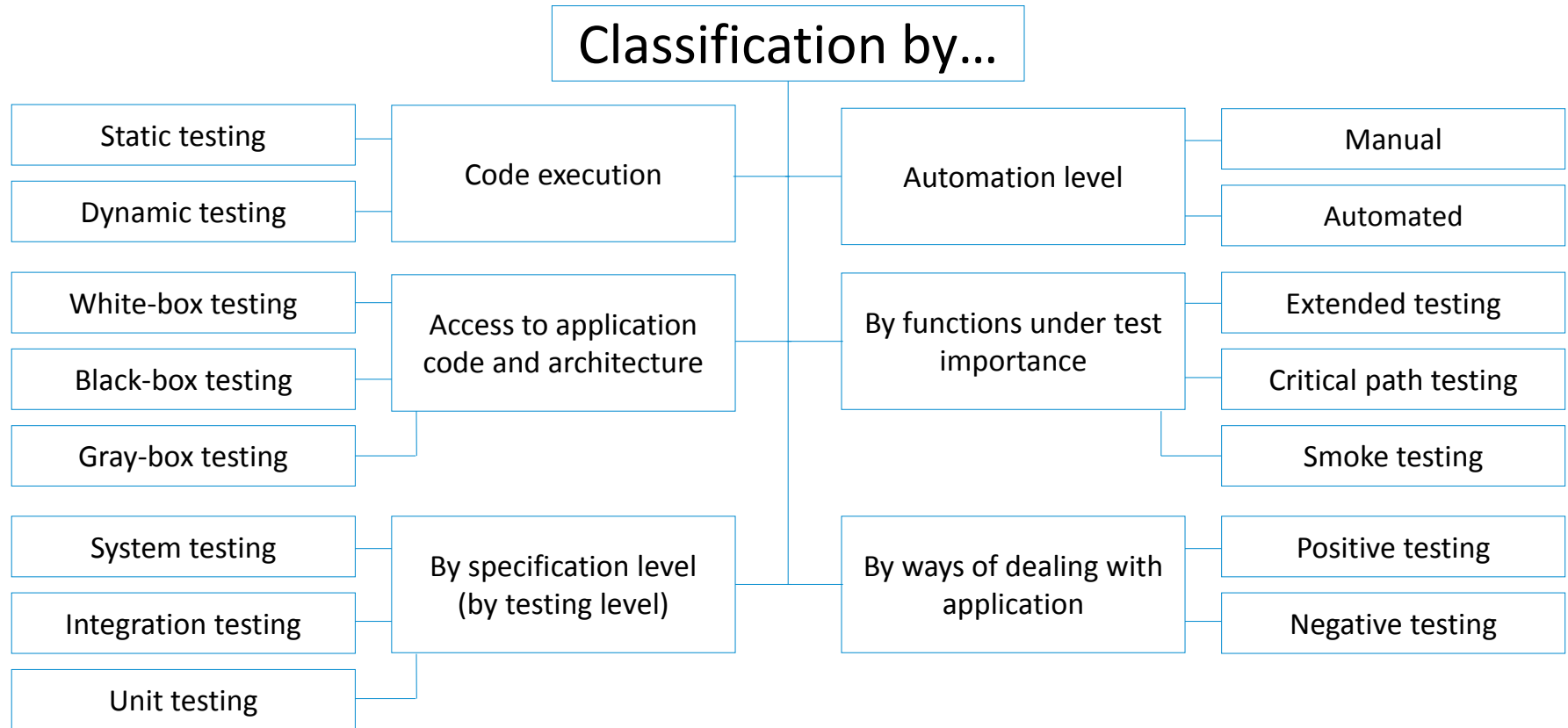
Functional testing – testing conducted to evaluate the compliance of a component or system with functional requirements.

Functional requirement – a requirement that specifies a function that a component or system must be able to perform.

Non-functional testing – testing conducted to evaluate the compliance of a component or system with non-functional requirements.

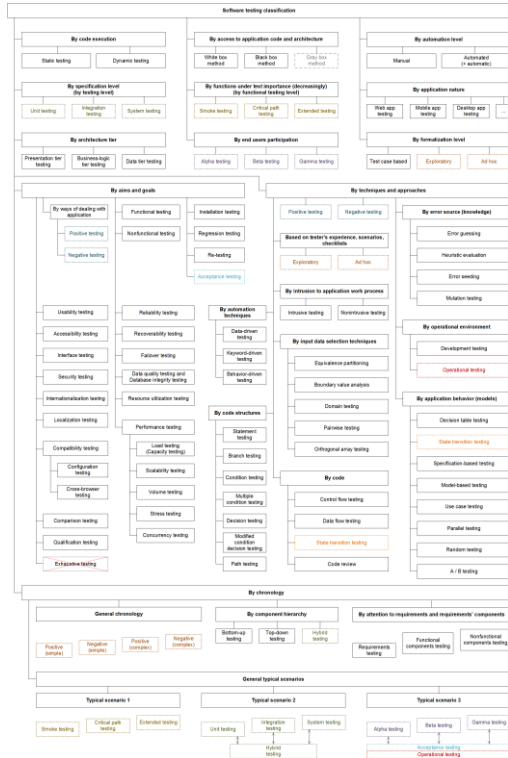
Non-functional requirement – a requirement that describes how the component or system will do what it is intended to do.

General Classification Approach

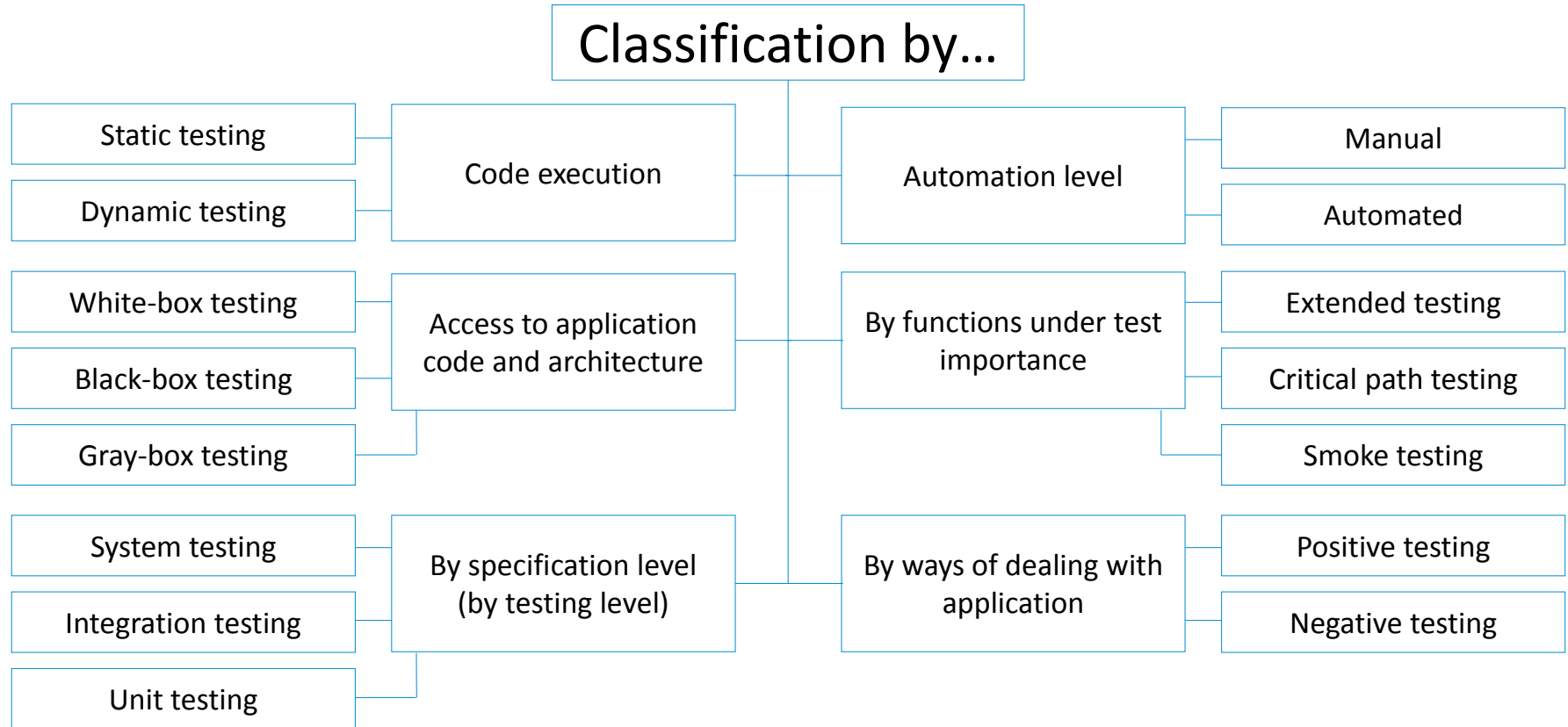


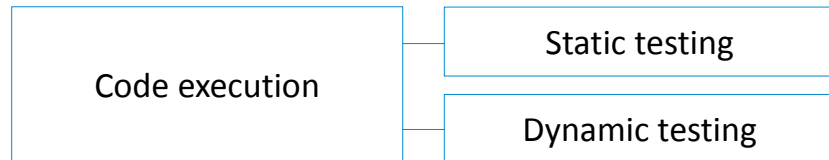
General Classification Approach: There Is Much More!

Please, visit:
http://svyatoslav.biz/urls/stc_en/



Classification by code execution



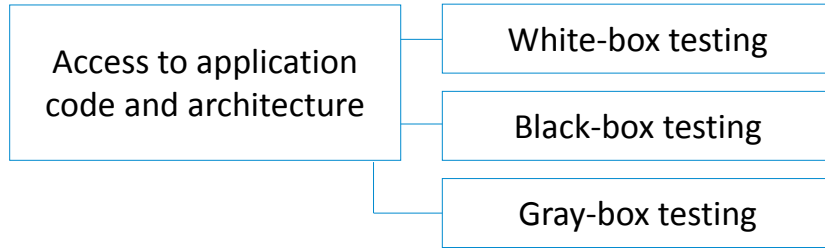


Static testing – testing a work product without code being executed.

Dynamic testing – testing that involves the execution of the software of a component or system.

... by access to application code ...

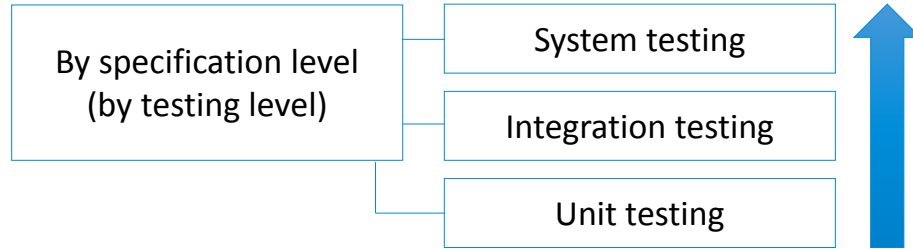
Read and remember!



White-box testing – testing based on an analysis of the internal structure of the component or system.

Black-box testing – testing, either functional or non-functional, without reference to the internal structure of the component or system.

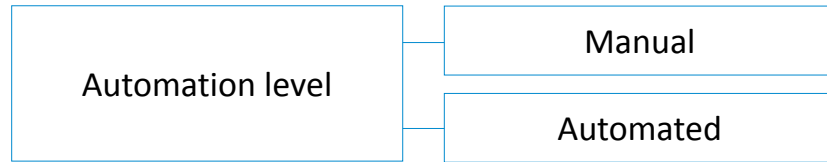
Gray-box testing – testing, which is a combination of Black-box and White-box (i.e. the internal structure is partially known).



Unit testing – the testing of individual hardware or software components.

Integration testing – testing performed to expose defects in the interfaces and interactions between integrated components or systems.

System testing – testing an integrated system to verify that it meets specified requirements.

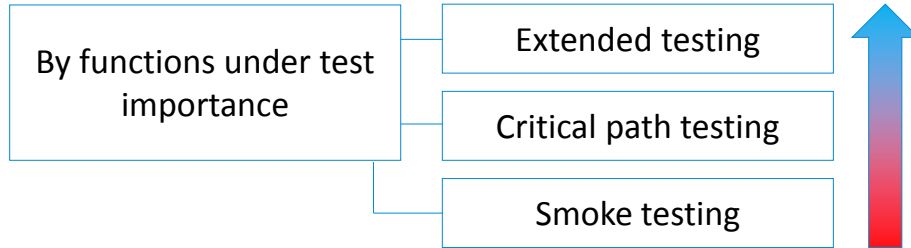


Manual testing – testing performed by the tester who carries out all the actions on the tested application manually.

Automated testing – the use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions.

... by functions under test importance

Read and remember!



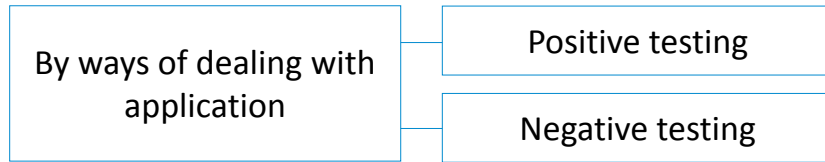
Smoke test – a subset of all defined test cases that cover the main functionality of a component or system, the most crucial functions.

Critical path test – test cases that cover the functionality used most of the time by the majority of users.

Extended test – test cases that cover the “nice-to-have” functionality (not used most of the time by the majority of users).

... by ways of dealing with application

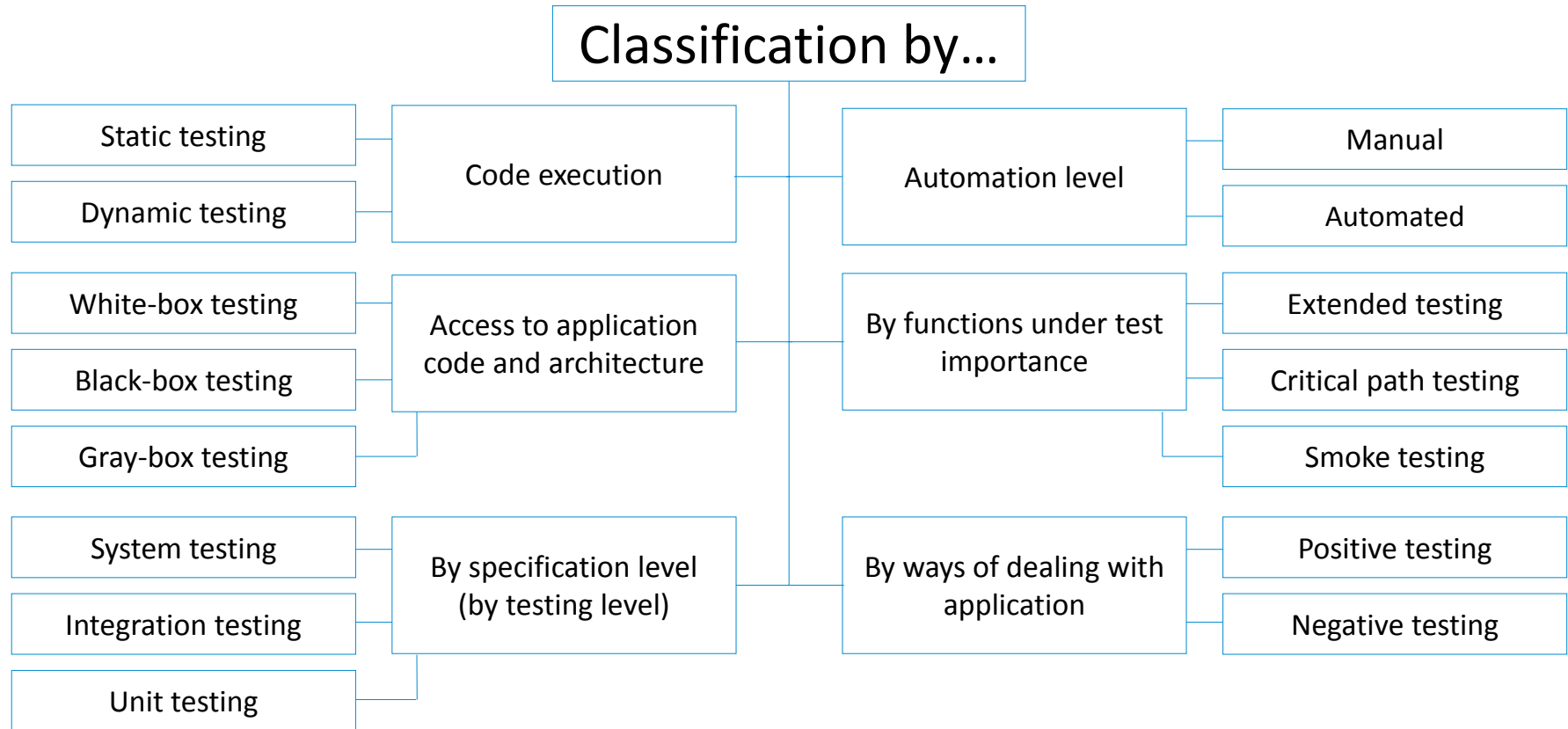
Read and remember!



Positive testing – testing process where the system validated against the valid input data and valid set of actions.

Negative testing – tests aimed at showing that a component or system does not work.

Let's look at the big picture one more time...





Software Testing Classification

Software Testing Introduction



TRAINING
CENTER

