Software Testing

Training

Fabian Piau

2012-11-26

ALTRAN

Summary

1. Requirements
2. Theory & Models
3. ISTQB certification
4. Test management
5. Infrastructure & Tooling

Requirements analysis

- Define the list of requirements
  - done at the beginning of the project
- The requirements should be
  - documented, actionable, measurable, testable, traceable
  - related to identified business needs or opportunities
  - defined to a level of detail sufficient for system design
- Critical to the success of the project
  - Understand the customer needs

What is a requirement?

- Answers to the question: What I need?
  - DO NOT answer to: How to implement it?
- Must be structured
  - order by priority
- Must be atomic
  - High-level and detailed enough

Define the list of requirements

- 3 columns matrix

Requirements matrix		
Category	**Description**	**Priority**
Different categories	▪ Short description (keep it simple)	▪ High ▪ Normal ▪ Low

Requirement categories

- Functional, what a system must do
  - Based on business processes, use cases, scenario's, high-level description of system behavior
  - Core business features
- Non functional, about the system itself
  - Architectural
  - Performance
  - Confidentiality/Security
  - Reliability/Errors management
  - Usability
- Non-functional requirements are as important as functional requirements

Testing is all about context

- Some software projects
  - Personal blog
  - Government website
  - Airplane embedded software
- Not all projects have the same risk
- Testing is context dependent

Why is testing necessary?

- Aim
  - Reduce the risk
  - Improve the quality
    - On time
    - To budget
    - To specification
  - Meet the contractual requirements
  - Verify legal and industry specific standards

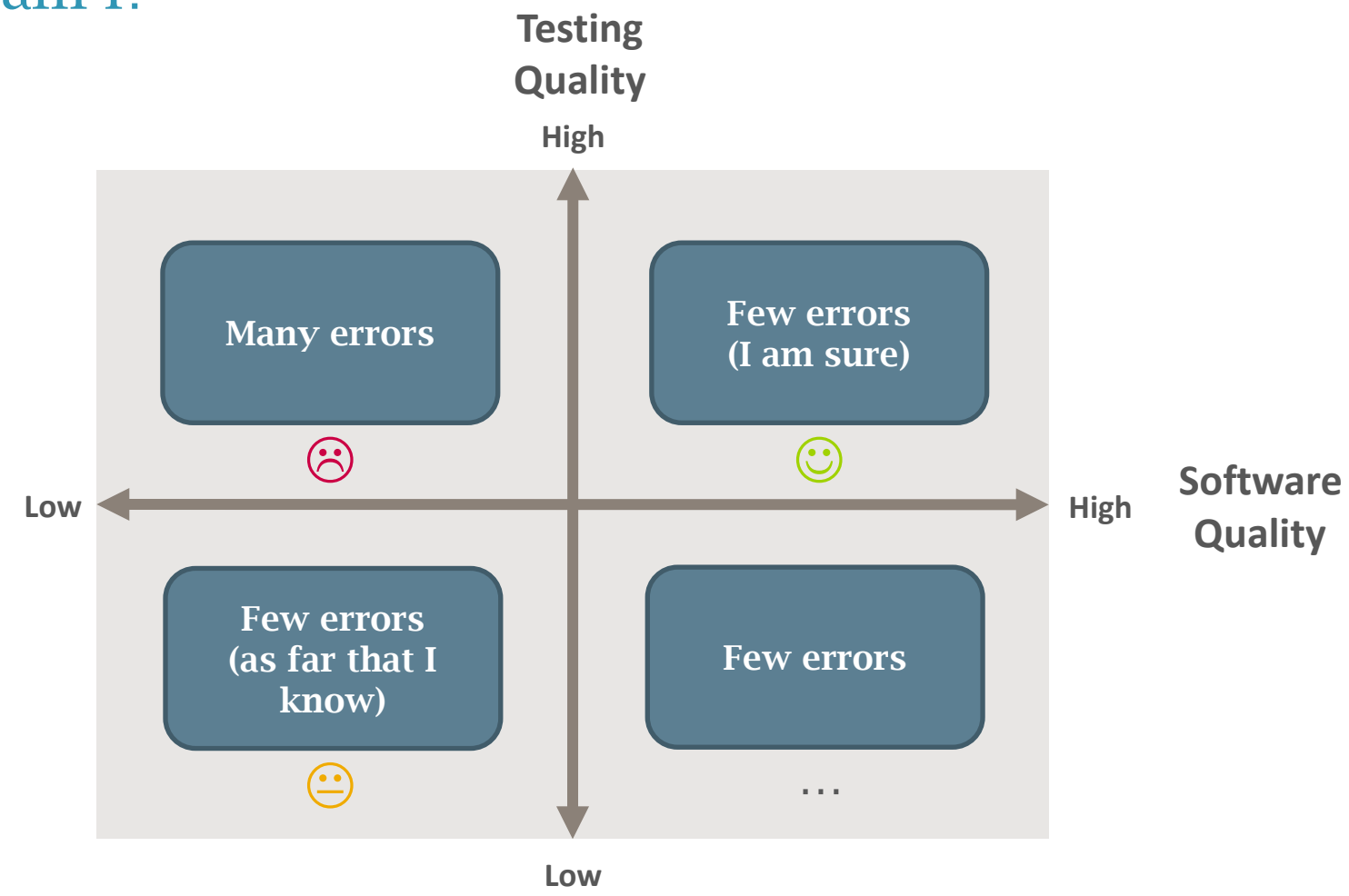
- Quality is a measurement of
  - Functionality (adequacy, interoperability, correctness, security…)
  - Reliability
  - Usability
  - Efficiency
  - Changeability, portability (e.g.: easiness to create software updates)
  - Maintainability (e.g.: easiness to install)

Testing = Improve quality

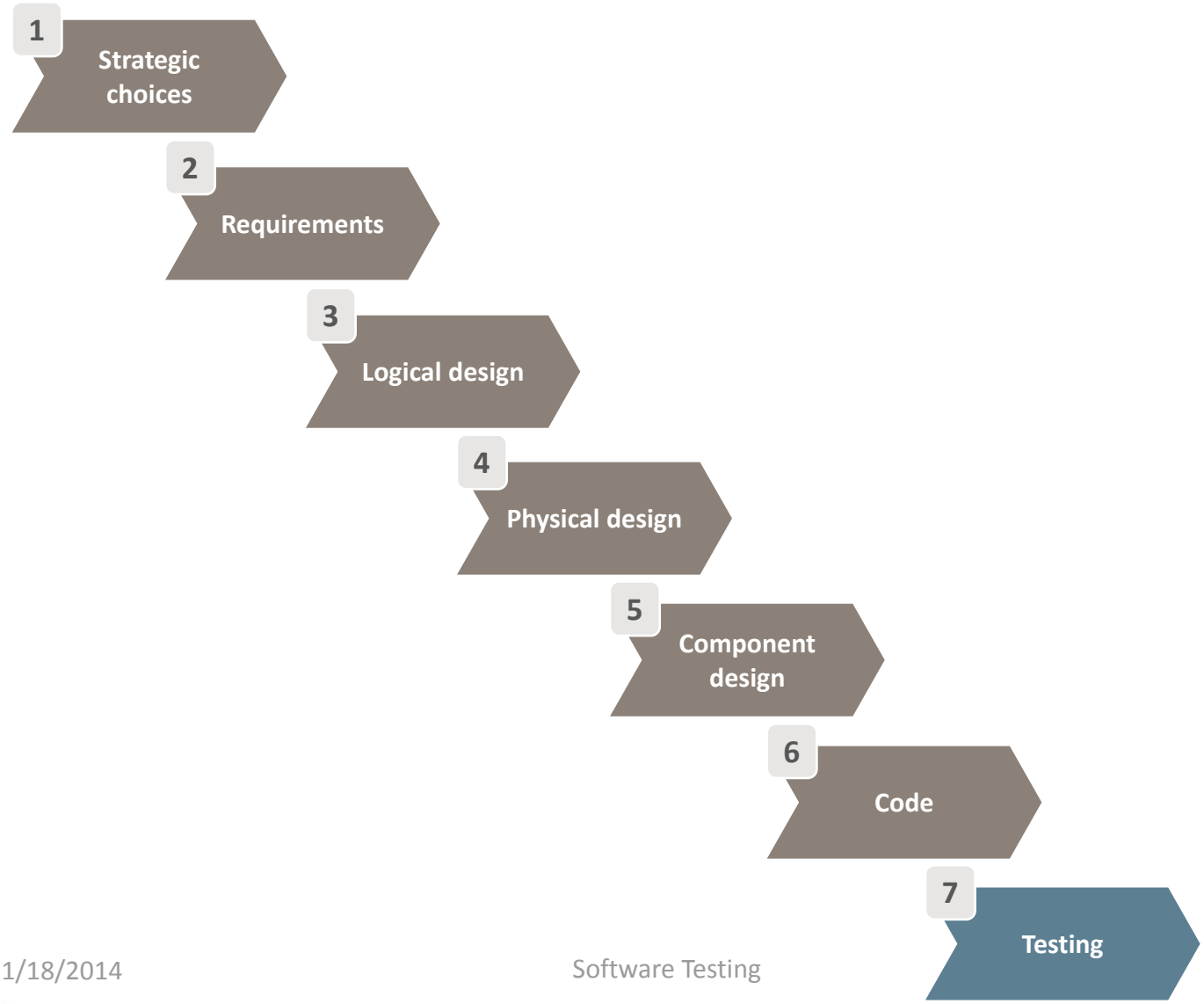
- Quality increases when
  - Defects are found
  - Defects are solved and re-tested

- Exhaustive testing is impossible
  - Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases.
  - Instead of exhaustive testing, we use risk and priorities to focus testing efforts.
  - Testing can show that defects are present, but cannot prove that there are no defects.
  - Testing reduces the probability of undiscovered defects remaining in the software but even if no defects are found, it's not a proof of correctness.

Where am I?


Testing Quality
High
Low
Many errors
Few errors (I am sure)
Few errors (as far that I know)
Few errors
...
Low
High
Software Quality

The waterfall model

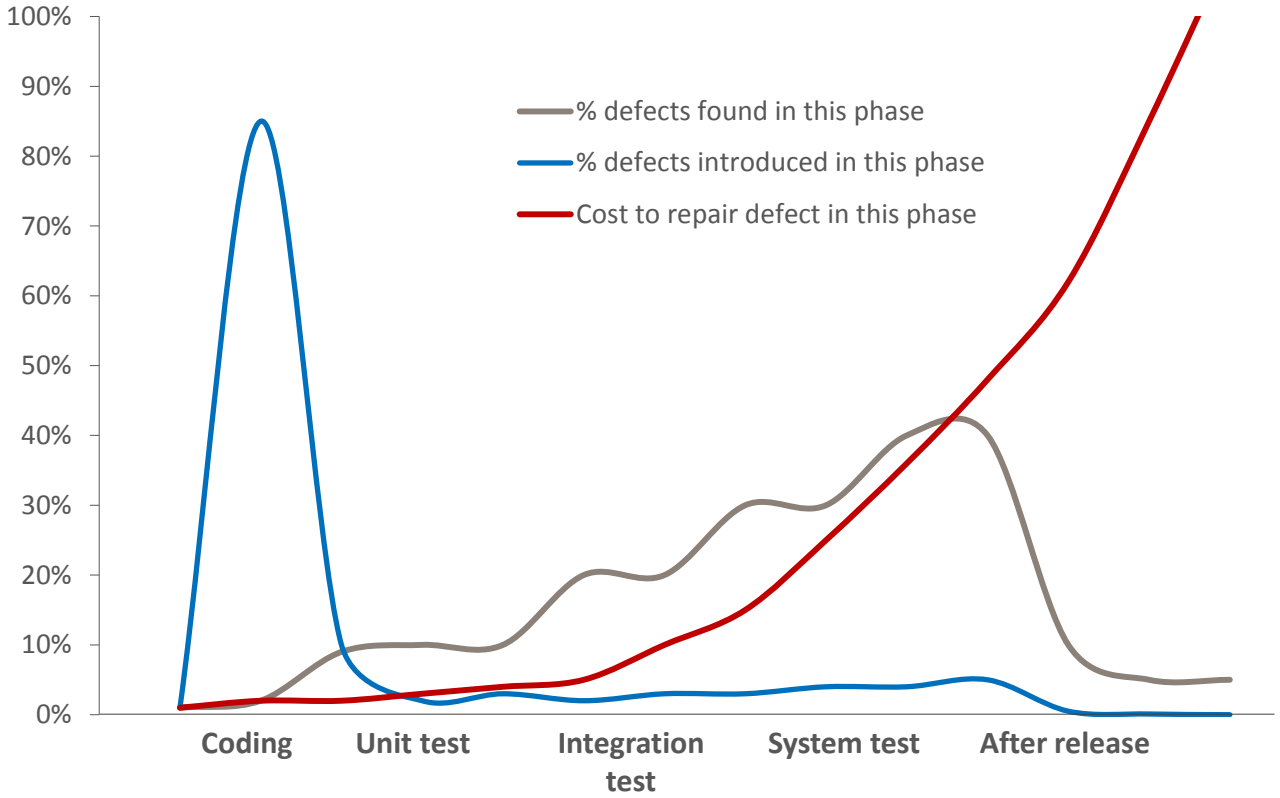


The waterfall model

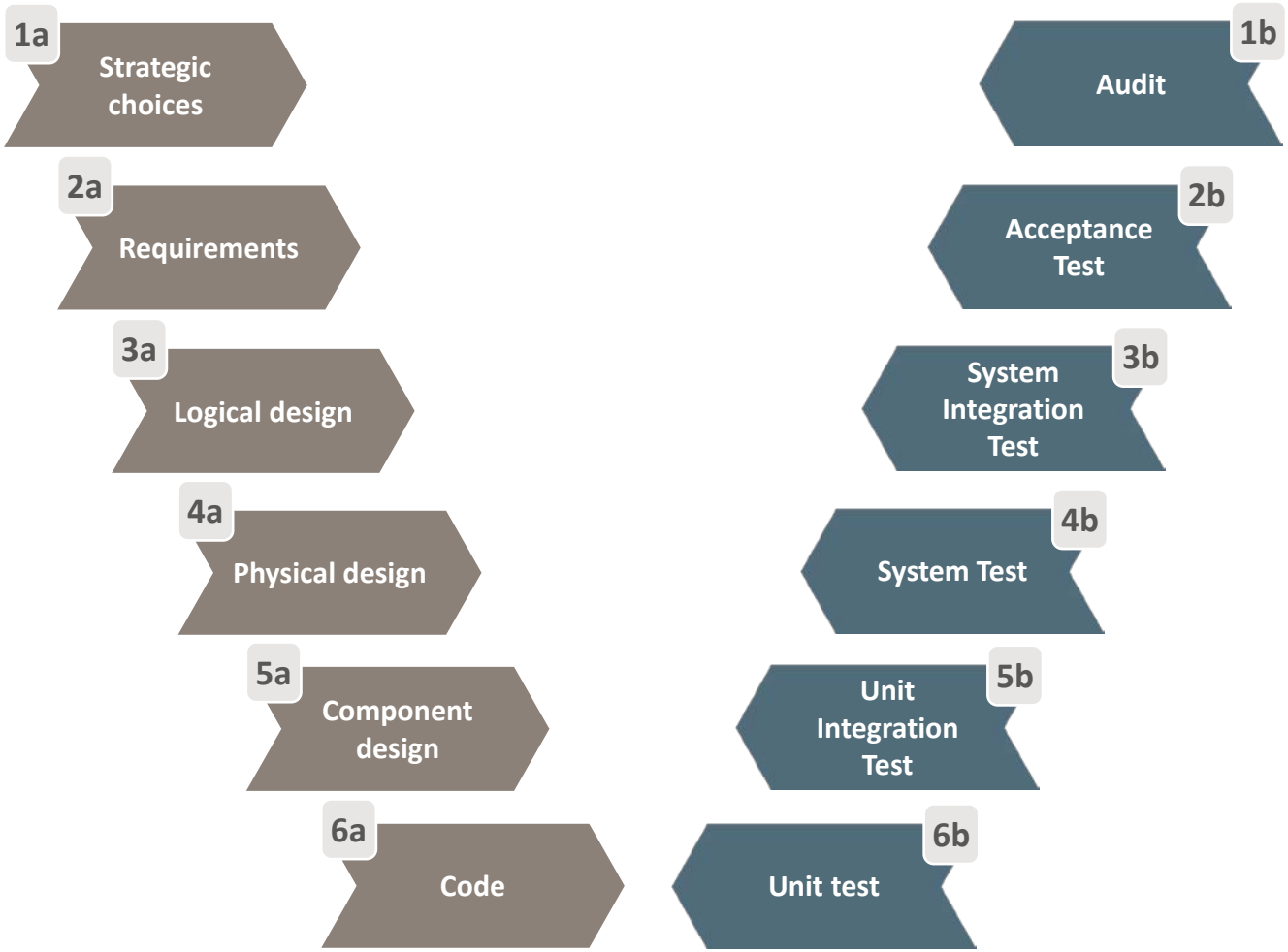
- Sequential and linear
- Ok in the 70s, but outdated now
- Testing is done at the end
  - Too late

The cost of defect

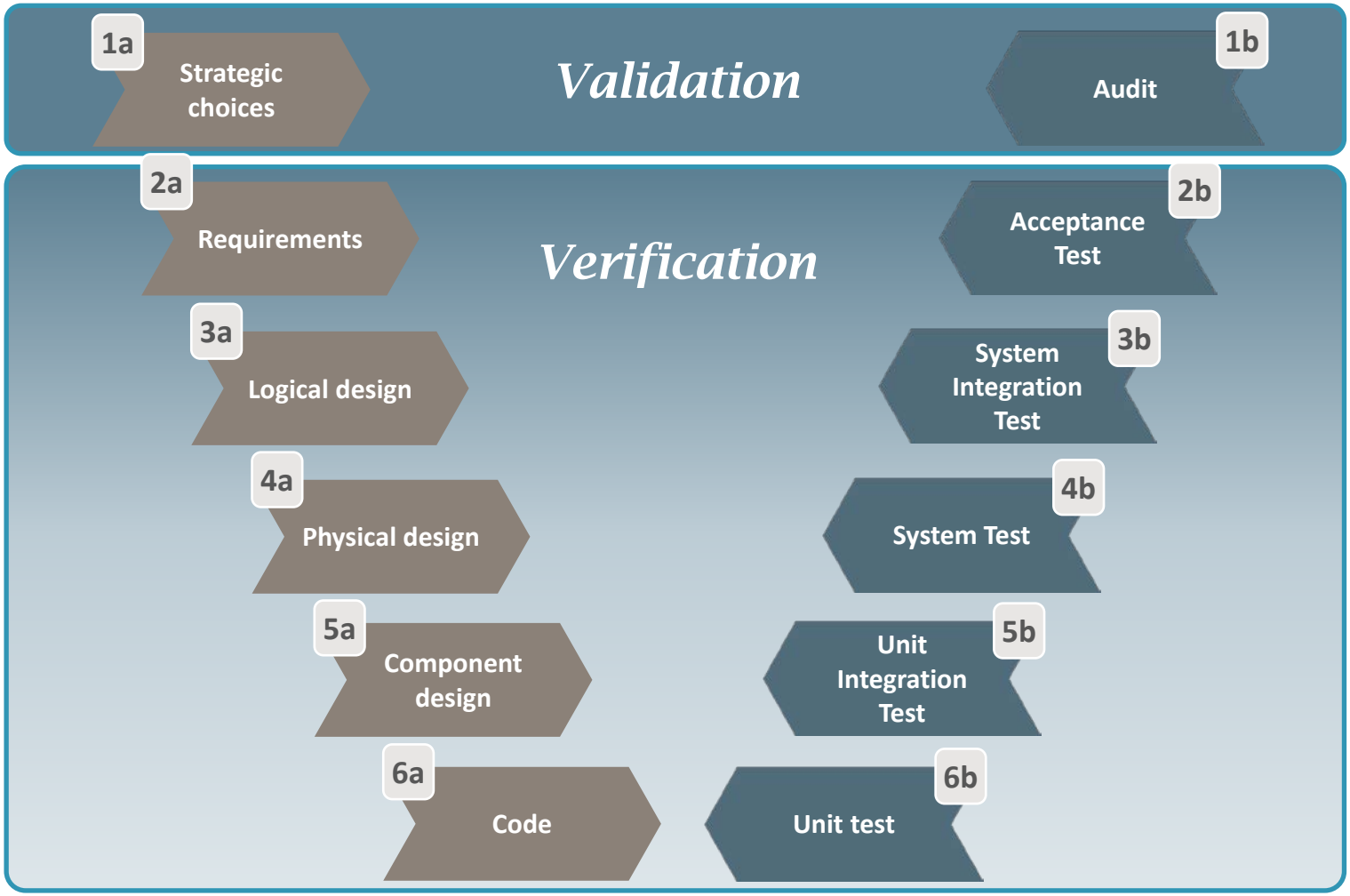
Cost of defect



The V model


1a
Strategic choices
2a
Requirements
3a
Logical design
4a
Physical design
5a
Component design
6a
Code
1b
Audit
2b
Acceptance Test
3b
System Integration Test
4b
System Test
5b
Unit Integration Test
6b
Unit test

The V model


1a
Strategic choices
Validation
1b
Audit
2a
Requirements
Verification
2b
Acceptance Test
3a
Logical design
3b
System Integration Test
4a
Physical design
4b
System Test
5a
Component design
5b
Unit Integration Test
6a
Code
6b
Unit test

The V model

- Sequential
- Testing through the whole process
  - Dedicated testing for each phase
  - Development & testing are equal important
- Early testing
  - Testing activities should start as early as possible in the software or system development life cycle and should be focused on defined objectives

Different test levels

- Unit testing (also know as component testing)
  - What?
    - Search for defects in the code. Verify the functionality of a specific section of code (e.g. modules, objects, classes, methods, etc.) that is separately testable
  - Who?
    - Tester needs knowledge of development languages → Developers
  - How?
    - xUnit
    - Mocks (for isolation)

Different test levels

- Integration testing
  - What?
    - Search for defects in the interactions between integrated components or systems. Verify the interfaces between components.
  - Who?
    - Tester needs knowledge of development languages → Developers
  - How?
    - xUnit

Different test levels

- System testing
  - What?
    - Search for defects in an integrated system. Verify that it meets specified (functional and non-functional) requirements (= Testing the behavior of the whole system)
  - Who?
    - Tester without knowledge of the implementation
    - Often carried out by independent test team
    - Previous test levels tested with technical viewpoint, now we're more end-user/customer related
  - How?
    - Test environment as close as possible to production environment

Different test levels

- Acceptance testing
  - What?
    - 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 (= Establishing confidence in the software)
  - Who?
    - Customer or end user
  - How?
    - Test environment as close as possible to production environment

When should I stop to test?

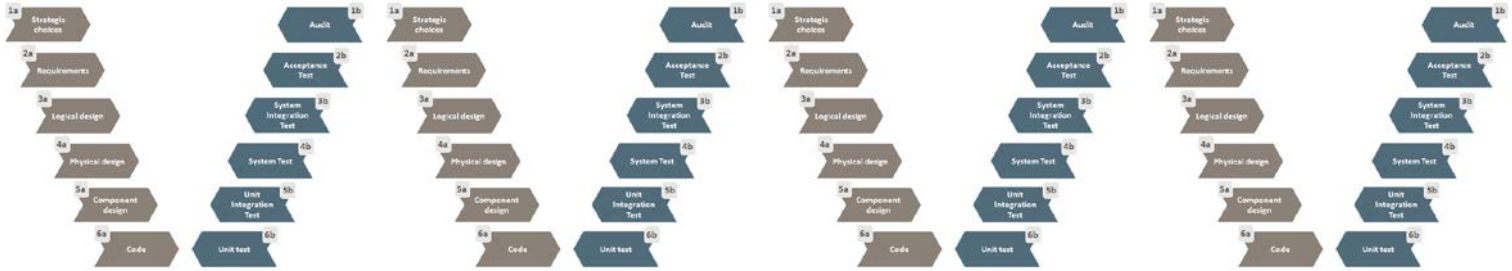
- Define and agree upfront (in test plan) the exit criteria
  - The set of generic and specific conditions, agreed upon with stakeholders for permitting the test process to be officially completed

- Bad criteria
  - If too low: Releasing bad software
  - If too high: Not reachable, expensive
  - Examples:
    - Time has run out
    - Money has run out
    - The software must be delivered
    - No errors found at all

- Good criteria
  - The (critical) requirements have been met
  - The required test techniques have been applied
  - “Enough” error have been found. (not absolute!) (error seeding)
  - Mean Time Before Error (MTBE) has dropped under predefined threshold
  - Examples:
    - For all critical requirements tests have been designed and executed.
    - Pass percentage is 100% for all critical requirements.
    - All high-severity defects have been solved and re-tested.
    - There is an agreement between all involved parties about the list of outstanding defects.

Agile model

- Iterative-incremental model
- Small increments with minimal planning and do not directly involve long-term planning
  - "Mini"-V models series



- Continuous (integration, delivery)
- Modern testing approach (TDD)

ISTQB certification

- Give the vocabulary about software testing
- Understand the main concepts

- Definition of testing: The **process** consisting of **all the life cycle activities**, both **static and dynamic**, 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 errors.**

- Static testing:
  - Testing an object without the execution of the software
  - Examples: walkthrough, inspections, static analysis

- Dynamic testing:
  - Testing that involves the execution of the software or a component or system
  - Examples: leak test, stress tests, performance tests…

Error, defect or failure?

- **Error** (mistake)
  - Human action that produces an incorrect result
  - In code or documents
- **Defect** (bug, fault)
  - A flaw in a component or system that can cause the component or system to fail to perform its required function. A defect if encountered during execution, may cause a failure of the component or system.
  - Can occur because of human beings are fail able and because there is time pressure, complex code, complexity of infrastructure, changed technologies and/or many system interactions.
  - Present since it was developed or changed. (has it roots in the developed software)
- **Failure**
  - Deviation of the component or system from its expected delivery, service or result.
  - Can be caused by environmental conditions. (radiation, magnetism, electronic fields, pollution)
  - Environmental: Also changes in hardware conditions.
  - A not fulfilled requirement.

Static analysis

- What?
  - Manual examination (reviews) of all documents of value to the software project (contracts, program code, test plans, manuals…)
  - Automated analysis (static analysis)
- In contrast with dynamic analysis
  - Needs to execute the code
  - But has the same objective as dynamic analysis
- Benefits
  - Early detection and correction of problems.
  - Finding of bugs which can't be found by dynamic methods

    Like:
    - Finding causes of failures
    - Omissions in requirements
    - Deviations from standards
    - Design defects
    - Insufficient maintainability
  - Early warning about suspicious aspects of the code or design (calculating metrics)
  - Detecting dependencies and inconsistencies
  - Improve maintainability
  - Prevention of defects
  - Discussion and decision by consensus
  - Gain understanding
- Done by using a team of people
  - Leads to mutual learning
  - Whole team feels responsible for the quality of the examined object

Static analysis – Manual reviews

	Process	Audience	Lead	Focus	Specific	Purpose
Informal Review	No formal process	-	-	Design and code	▪ Pair programming ▪ Reviewing code	Quick benefit
Walkthrough	Informal to formal	Peers	Author	Work product	▪ Scenarios ▪ Dry runs	▪ Learn ▪ Understand ▪ Find defects
Technical Review	▪ Documented ▪ Informal to formal	▪ Peers ▪ Technical experts	Moderator	Work product	▪ Checklists ▪ Review report	▪ Discuss ▪ Decide ▪ Evaluate alternatives ▪ Find defects
Inspection	Formal	Peers	Moderator	▪ Work product ▪ Process	▪ Process based ▪ Entry/exit criteria ▪ Roles ▪ Metrics	Find defects

Type of review should be selected based on requested quality and the effort to spent.

Static analysis – Automated tools

- To discover
  - Referencing/reading of undefined variables
  - Inconsistent interface between modules
  - Variables never used
  - Unreachable code
  - Programming standard violations
  - Security vulnerabilities
  - Syntax violations of code and software models
- Examples of tools (java)
  - Compiler, checkStyle, findbug…
- Example of metrics
  - Lines of code (LOC)
  - Nested levels
  - Number of function calls
  - Comment frequency
  - Cyclomatic complexity (Mc Cabe 1976 – complexity within one module)

Dynamic analysis - Test design techniques

- Black-box
  - Procedure to derive and/or select test cases based on an analysis of the specification, either functional or non-functional, of a component or system without reference to its internal structure
  - No knowledge about implementation
- White-box
  - Procedure to derive and/or select test cases based on an analysis of the internal structure of a component or system
  - Knowledge about implementation
- And also experienced-based testing
  - Anticipate what defects might be present
  - Error Guessing & Exploratory Testing

Test types

- Functional testing
  - Testing « what » the system does
- Non-functional testing
  - Testing the attributes of the functional behavior
- Testing related to changes
  - Confirmation testing. Retesting to confirm that original defect has been successfully removed.
  - Regression testing. Testing of a previously tested program after modification to ensure that defects have not been introduced in unchanged areas of the software.

Regression testing

- Regression (or non-regression) testing is verifying that all the unchanged parts of a system still function correctly after the implementation of a change.

- Regression testing is not retesting
  - It is the complementary.

- Different strategies
  - Rerun all tests that have detected faults
  - Rerun all tests that cover changed parts
  - Rerun all tests that cover newly integrated parts
  - Rerun all tests

Test organization

- Roles in a testing team are very similar to roles present in a development team
- Test manager
  - Test planning and test control expert, possessing knowledge and experience in the fields of software testing, quality management, project management and personnel management.
- Tester
  - Expert in creating and executing tests, test setup incident reporting. (IT knowledge, testing basics, applying test tools, understands the test object)
- Other possible roles (sub division of Tester)
  - Test developer: is responsible for the technical aspect of the automated test suite. He translates the design of the test suite to modules in the test tool to be realized.
  - Test designer: designs the logical and physical test cases.
  - Test performer: executes the logical and physical test cases.
- Additional roles
  - Test Program Manager: is responsible of the testing improvement process. He guarantees that testing in the company/department/service is a mature process.
  - Test Methodology Expert: is a specialist in being a tester. He is experienced and fulfills a combination of executing and advisory roles in projects.

Test organization & independence

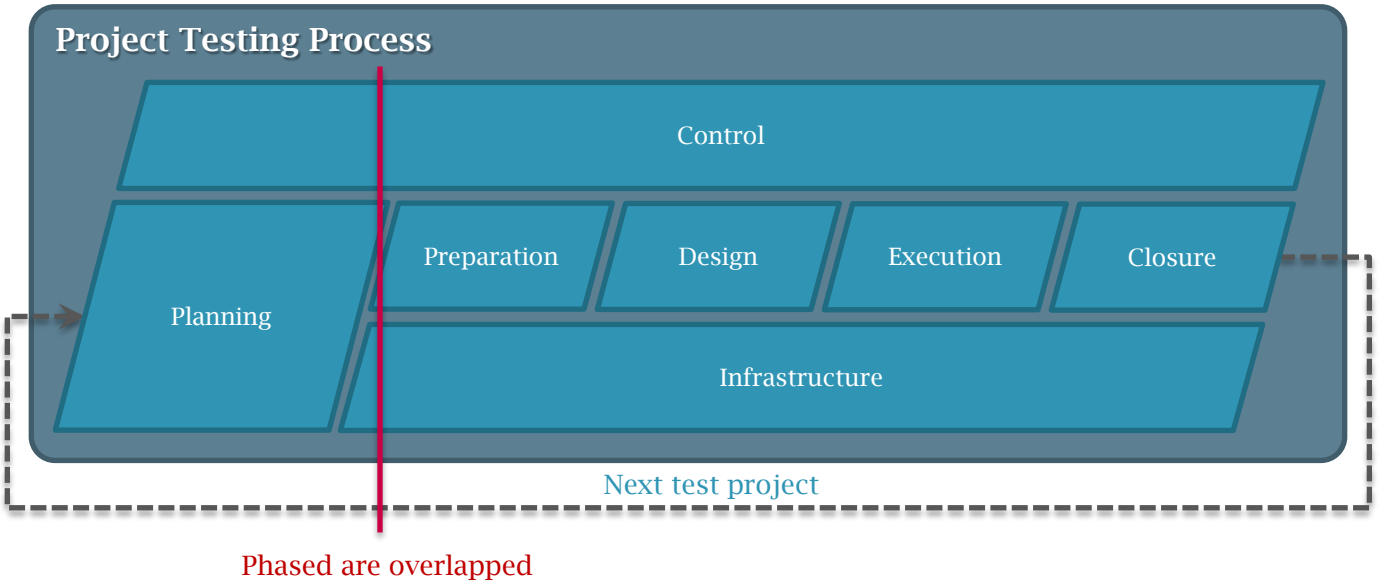
- Independent testing is more effective
  - Unbiased testers see other and different types of defects
  - Developers lose sense for responsibility for quality (difficult to destroy your own work)

- Options of independent testers:
  - (independent) tester within development team
  - Independent test team in organization or development
  - Independent testers from business organization
  - Independent test specialists
  - Dedicated external test team (outsourced, other department)

Test process

- The testing process inside a project is composed of seven phases:
  - planning (definition of the scope, the strategy, the budget, the participants/roles, the schedule, the exit criteria…)
  - preparation (gathering all necessary inputs to start design)
  - design (writing of test cases)
  - execution (execution of the test cases and defects follow-up)
  - control & monitor (reporting, KPI's…)
  - infrastructure (set up of the necessary tooling to support the testing process)
  - close & evaluate (collect data results, consolidate experience, lessons for future…)

Test process



Configuration management

- In testing, configuration management ensures that:
  - All test items are identified, version controlled, tracked for changes, related to each other and development items (test objects) so that traceability can be maintained throughout the test process.
  - All identified documents and software items are referenced unambiguously in test documentation.
- For the tester, configuration management helps to uniquely identify (and to reproduce) the tested item, test documents, the tests and the test harness.
- During test planning, the configuration management procedures and infrastructure (tools) should be chosen, documented and implemented.

Incident management

- Incidents:
  - Are discrepancies between actual and expected outcomes.
  - All discrepancies should be logged, but try to avoid duplicates.
  - Testers should not investigate the cause of a recorded incident. (= debugging)
  - Need tracking. (status-action workflow)
  - Should be classified. (incident classification)
  - Are raised under SDLC: development, review, testing.
  - Occur in code, tests and any type of documentation.

- Objectives of incident reports:
  - Provide developers and other parties with feedback.
  - Provide test manager a means of tracking the quality of the system under test and the progress of the testing.
  - Provide ideas for test process improvement.

Infrastructure

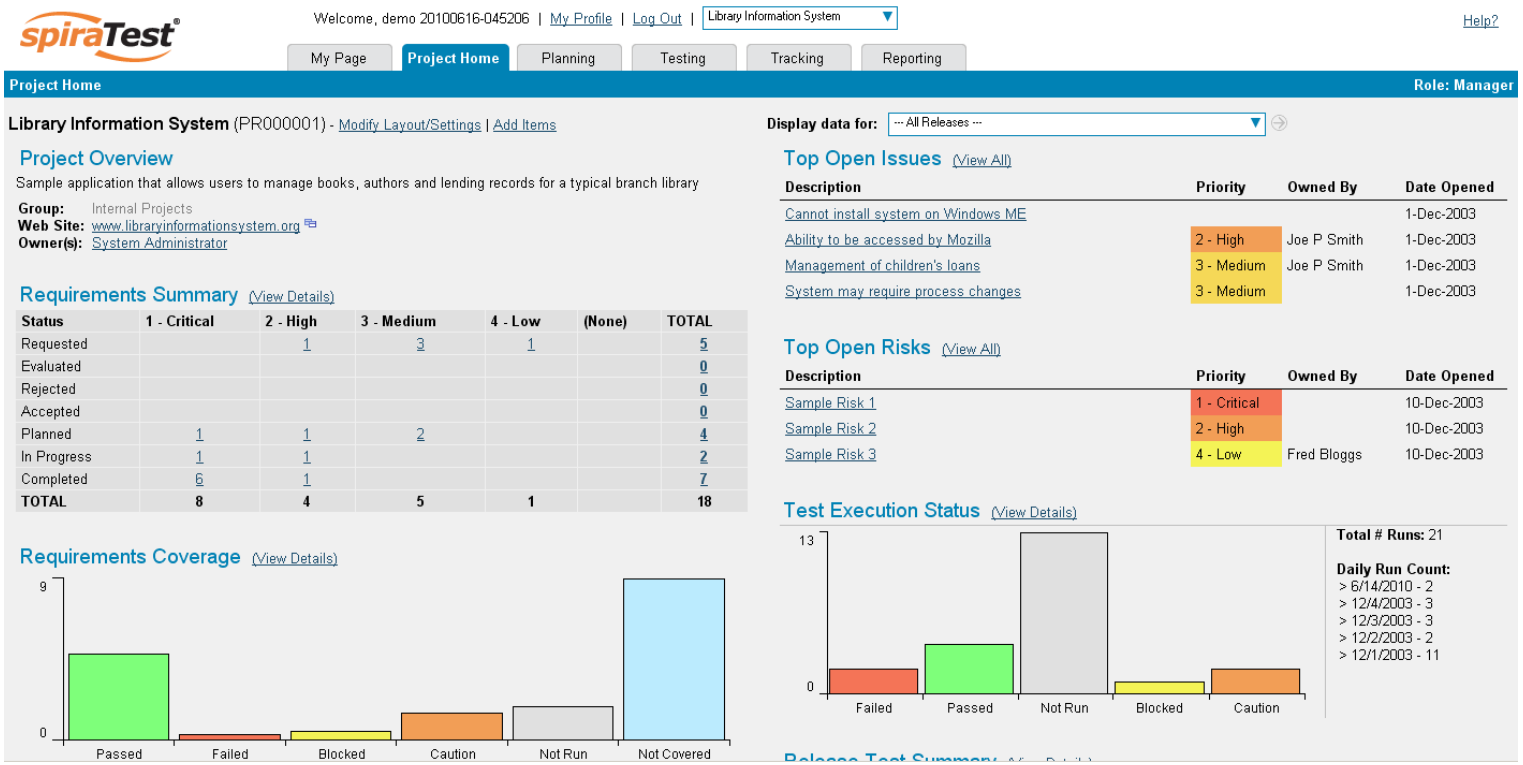
- Test infrastructure consists of the facilities and resources necessary to facilitate the satisfactory execution of the test. A distinction is made between test tools, test environments and workplaces.
- A test tool is an automated instrument that supports one or more test activities, such as planning, control, specification and execution.
- Test tools are classified as follow :
  - Tools for planning and controlling the test (HP Quality Center, TestLink, …)
  - Tools for designing the test (HP Quality Center, TestLink, …)
  - Tools for executing the test (HP Quality Center, TestLink, Bugzilla, Jira, Mantis, …)
  - Tools for debugging and analyzing the code

Automated testing & reporting

- Automated software test functions may be multiple :
  - Automated test case/data generation
  - Test case design from requirements or code
  - Selection of test cases
  - No intervention needed after launching tests
  - Set-up or recording of test environment
  - Run test cases
  - Captures relevant results
  - Compares actual with expected results
  - Report analysis of pass/fail
  - …
- Testing automation can :
  - Find important defects quickly
  - Measure and document product quality
  - Verify key features
  - Keep up with development
  - Assess software stability, concurrency, scalability…
  - Reduce testing costs
  - Reduce time spent in the testing phase
  - Improve testing coverage
  - Reduce impact on the bottom line
  - …

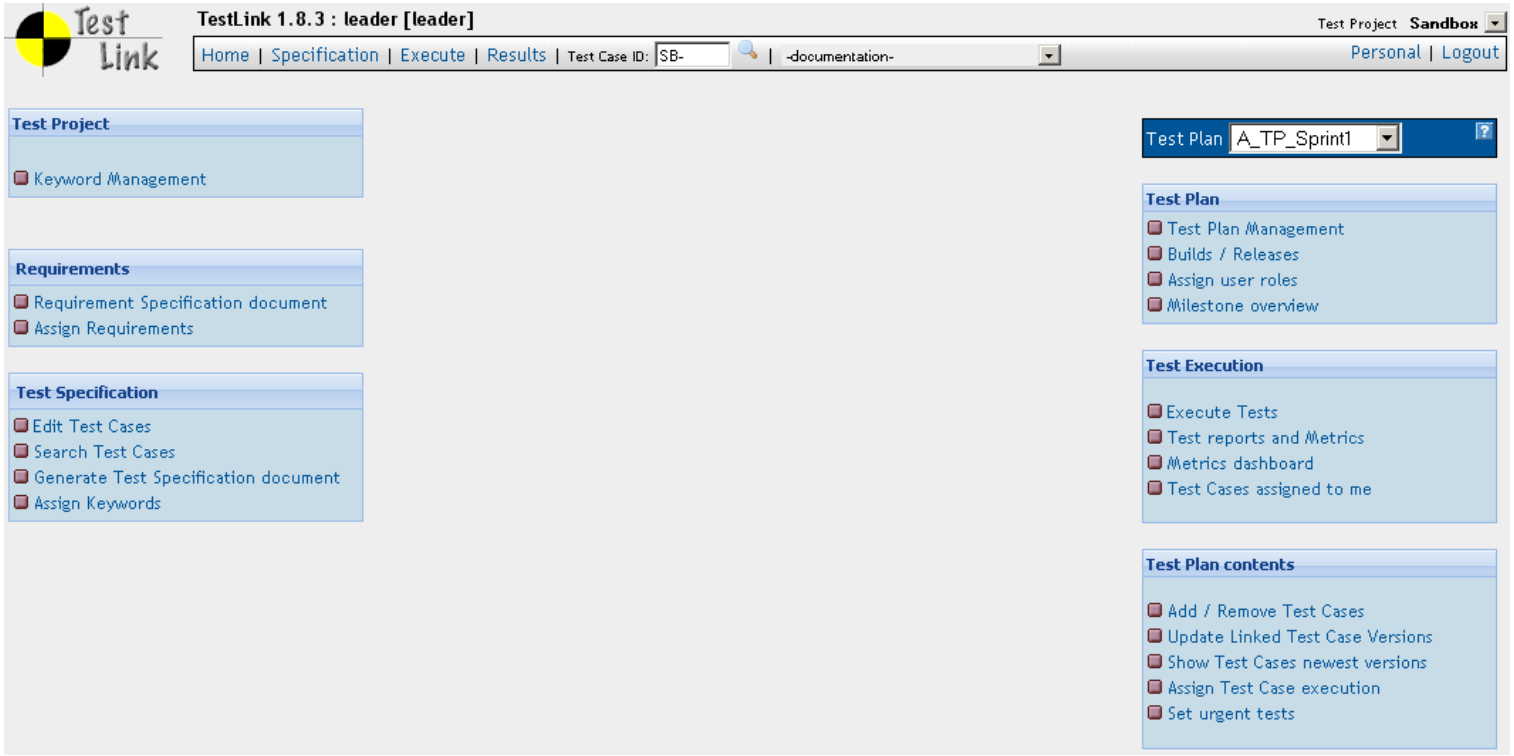
SpiraTest

- http://www.inflectra.com/SpiraTest/Demo.aspx



TestLink

- http://testlink.sourceforge.net/demo/login.php



Yes… but

- Purchasing or leasing a tool is no guarantee for success!
- It requires additional effort to achieve lasting results.

- Potential benefits
  - Repetitive work is reduced
  - Greater consistency and repeatability
  - Objective assessment
  - Ease of access to information about tests or testing

- Potential risks:
  - Unrealistic expectations for the tool.
  - Underestimating the time, cost and effort for the initial introduction of a tool.
  - Underestimating the time, cost and effort for continuing benefits of a tool.
  - Underestimating the effort required to maintain the test assets generated by the tool.
  - Over-reliance on the tool.
  - A tool never replace a nonexistent process or compensate for a sloppy procedure.

- Automating chaos just gives faster chaos !

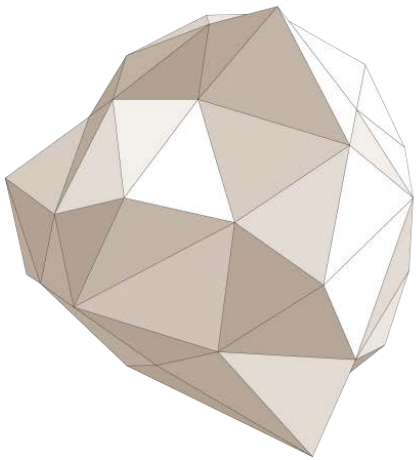
Some advices

- Tools are available for every phase of the test process, helping the tester to automate test activities or improve the quality of these activities.
- Use of a test tool is only beneficial when the test process is defined and controlled.
- Test tool selection must be a careful and well-managed process, as introducing a test tool may incur large investments.
- Information, training, and coaching must support the introduction of the selected tool. This helps to assure the future users' acceptance and hence the regular application of the tool.

Some last recommendations

- Steps that should be taken when selecting a test tool
  - 1. Requirement specification for the tool application.
  - 2. Market research. (Creating an overview of possible candidates.)
  - 3. Tool demonstrations and creation of short list.
  - 4. Evaluating the tools on the short list.
  - 5. Reviewing of the results and selection of the tool.
- Steps that should be taken when introducing a tool
  - 1. Execute a pilot project.
  - 2. Evaluate the pilot project experiences.
  - 3. Adapt the processes and implement rules for usage.
  - 4. Train the users.
  - 5. Introduce the tool in a stepwise fashion.
  - 6. Offer accompanying coaching.
- Introduce tools in following order:
  - 1. Incident management.
  - 2. Configuration management.
  - 3. Test planning.
  - 4. Test execution.
  - 5. Test specification.
- Take into account the learning curve !!

INNOVATION MAKERS



ALTRAN