Software Testing

Lesson 3
Testing in Software Life Cycles
V1.0

Uwe Gühl

Winter 2013 / 2014

Contents

- Testing in Software Life Cycles (1/2)
  - Definitions
    - Software quality
    - ISO/IEC 9126 Quality Model
    - Verification and Validation
    - COTS (Commercial Off-The-Shelf software)
  - Software Development Models
    - Sequential Development Model → Waterfall / V-Model
    - Iterative-incremental Development Models
    - Testing within a Life Cycle Model

Contents

- Testing in Software Life Cycles (2/2)
  - Test Levels
    - Component testing
    - Integration testing
    - System testing
    - Acceptance testing
  - Test Types
    - Functional Testing
    - Non-Functional Testing
    - Structural Testing
    - Re-testing and Regression Testing
  - Maintenance Testing

Definitions
Software quality

Discussion: What means (software) quality?

- "everyone feels they understand it" (Scott Pressman)
- Software quality characteristics (Steve McConnell)
  - external - those parts of a product that face its users,
  - internal - those that do not
- "a product's quality is a function of how much it changes the world for the better” (Tom DeMarco)
- "Quality is value to some person" (Gerald Weinberg)

Definitions
ISO/IEC 9126 Quality Model

- ISO/IEC 9126 Software engineering – Product quality [Wik14]
  - was an international standard for the evaluation of software quality – focusing on the product.
  - tries to develop a common understanding of the project's objectives and goals.
  - applies to characteristics to evaluate in a specific degree, how much of the agreements got fulfilled → Conformance level
- Hint: Since 2011 there is a successor available: ISO 25010-2011 has eight product quality characteristics (in contrast to ISO 9126's six), and 39 sub-characteristics

Definitions
ISO/IEC 9126 Quality Model

1 Functionality

2 Reliability

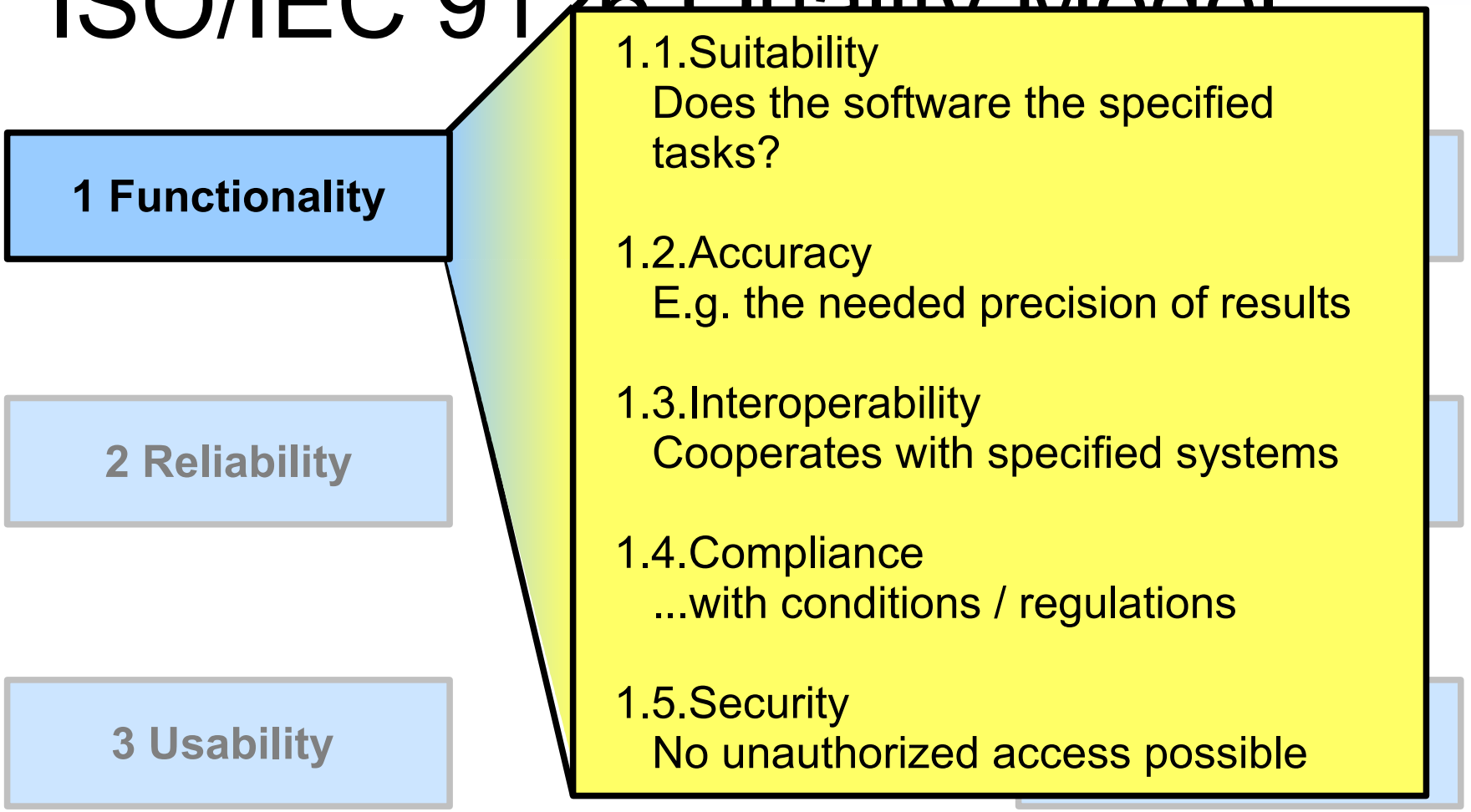
3 Usability

4 Efficiency

5 Maintainability

6 Portability

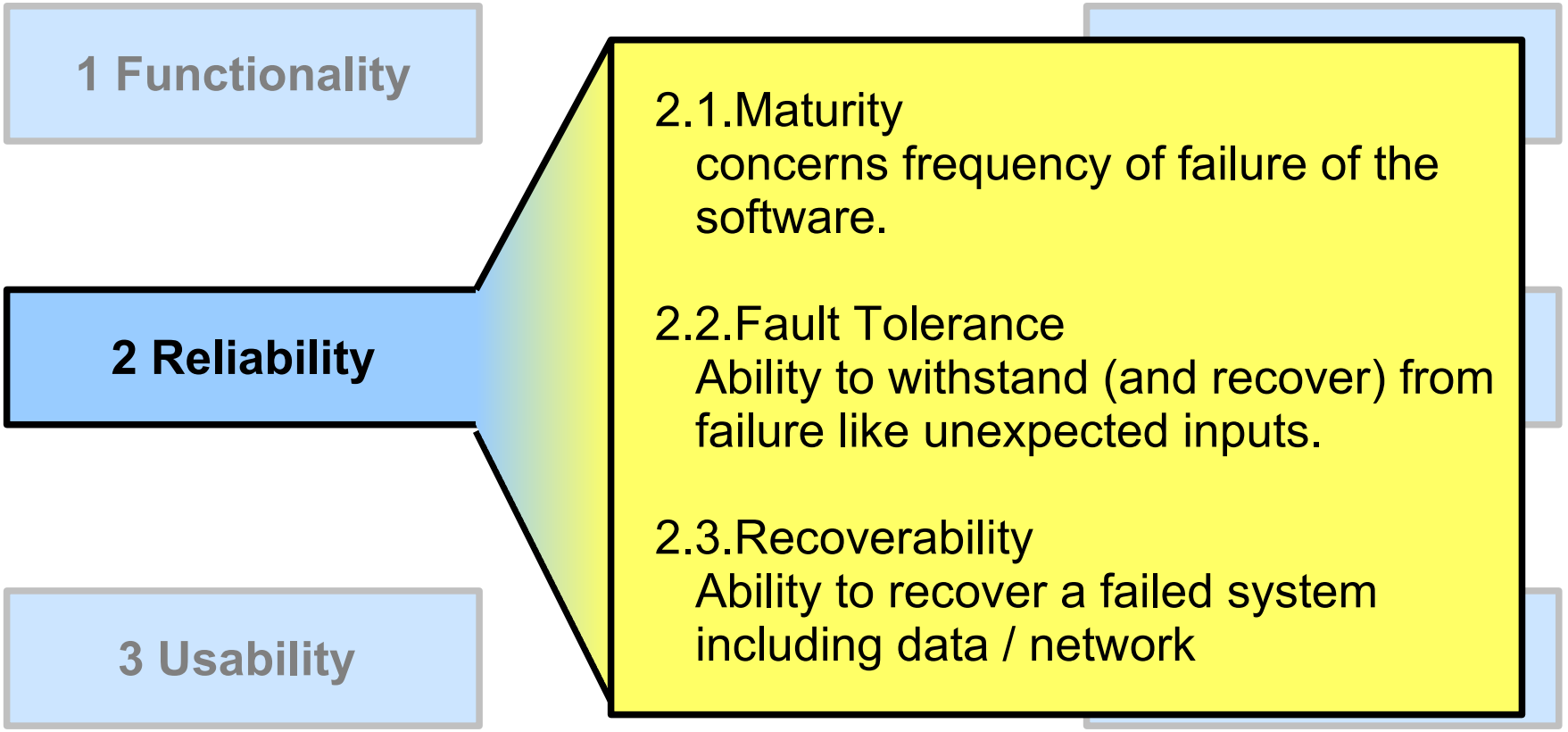
Definitions
ISO/IEC 9126 Quality Model



Definitions
ISO/IEC 9126 Quality Model

1 **Functionality** - A set of attributes that bear on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.
   - Suitability: Does the software the specified tasks?
   - Accuracy: e.g. the needed precision of results
   - Interoperability: cooperates with specified systems
   - Compliance: ...with conditions / regulations
   - Security: No unauthorized access possible

Definitions
ISO/IEC 9126 Quality Model

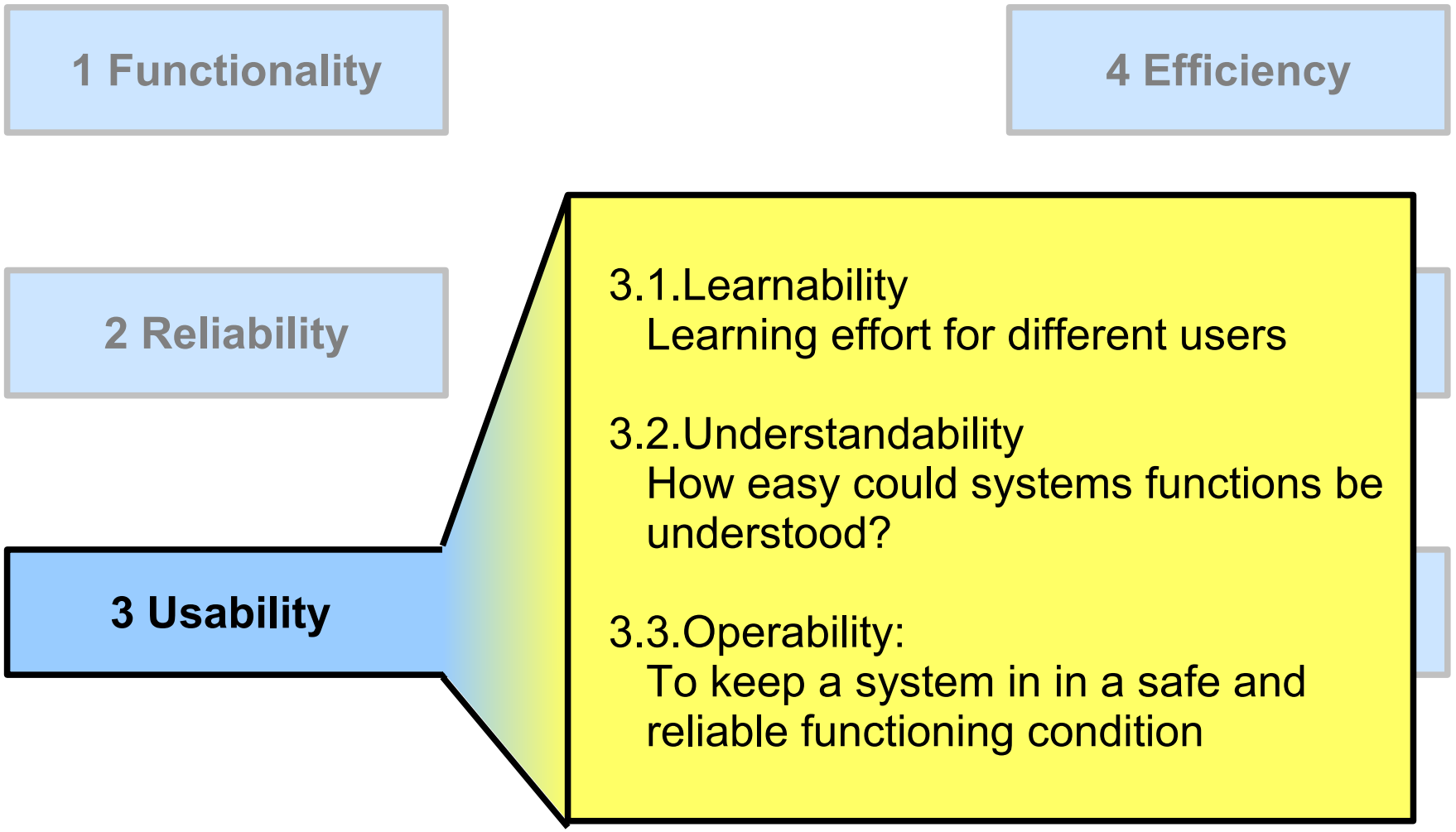


Definitions
ISO/IEC 9126 Quality Model

2 Reliability - A set of attributes that bear on the capability of software to maintain its level of performance under stated conditions for a stated period of time.

- Maturity: Minor breakdowns because of defects
- Recoverability: If there is a breakdown, how long does it need to recover – how much time / effort is needed (including data!)?
- Fault Tolerance: Can the system handle unexpected inputs?

Definitions
ISO/IEC 9126 Quality Model

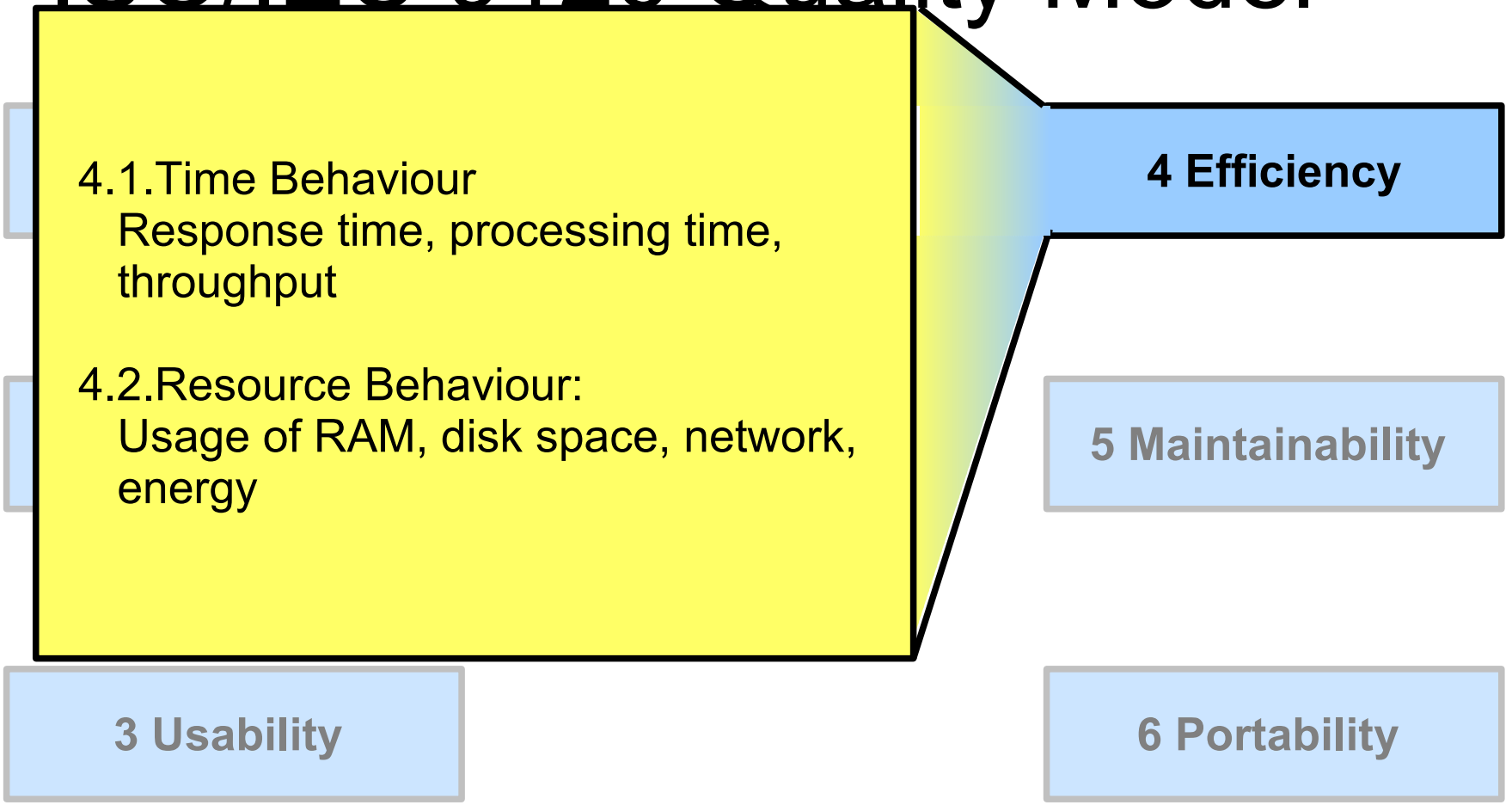


Definitions
ISO/IEC 9126 Quality Model

3 Usability - A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.

- Learnability: Effort to learn how to use a software
- Understandability
- Operability: To keep a system in in a safe and reliable functioning condition

Definitions
ISO/IEC 9126 Quality Model

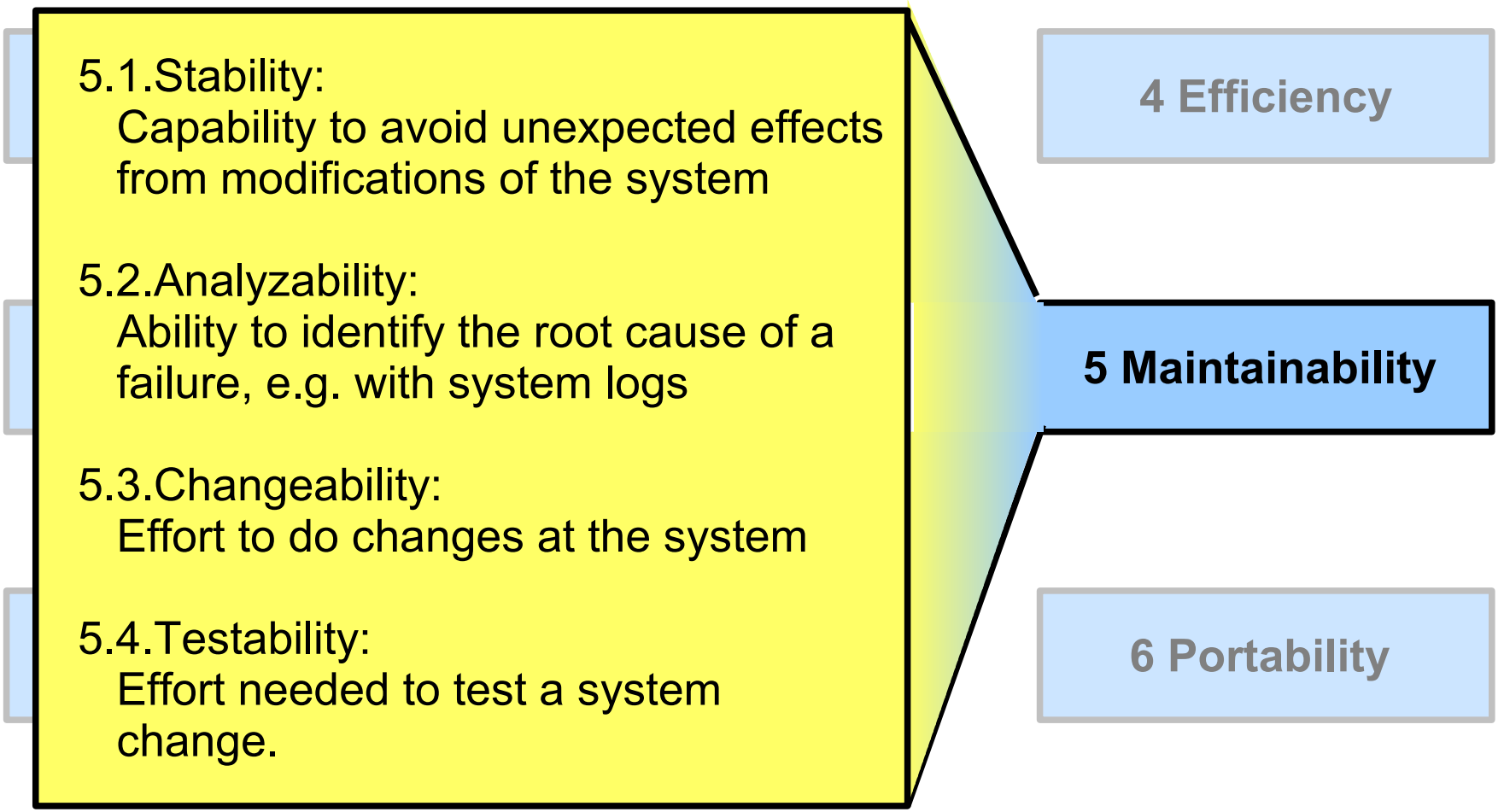


Definitions
ISO/IEC 9126 Quality Model

4 Efficiency - A set of attributes that bear on the relationship between the level of performance of the software and the amount of resources used, under stated conditions.

- Time Behaviour: Response time, processing time, throughput
- Resource Behaviour: Usage of RAM, disk space, energy

Definitions
ISO/IEC 9126 Quality Model

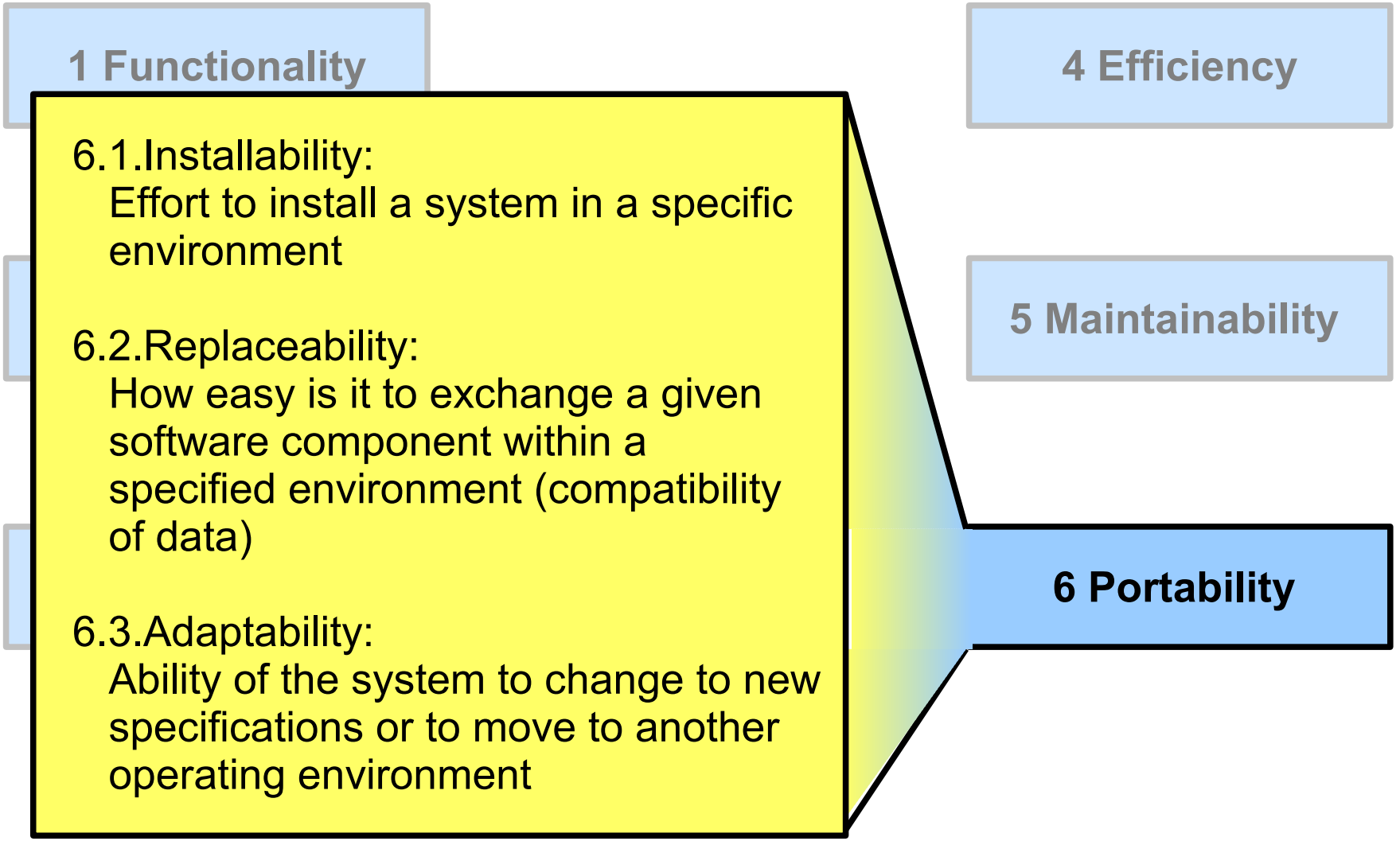


Definitions
ISO/IEC 9126 Quality Model

5 Maintainability - A set of attributes that bear on the effort needed to make specified modifications.

- Stability: What happens after a power cut?
- Analyzability: Monitoring the system
- Changeability: Changes at runtime possible?
- Testability: E. g. is it possible to reproduce activities?

Definitions
ISO/IEC 9126 Quality Model



Definitions
ISO/IEC 9126 Quality Model

6 Portability - A set of attributes that bear on the ability of software to be transferred from one environment to another.

- Installability: Effort to install a system in a specific environment
- Replaceability: With a specific different software (compatibility of data)
- Adaptability: E. g. move on another operating system

Definitions Verification & Validation

- Verification
  Did we build the right product?
  Verification is defined as the "demonstration of consistency, completeness, and correctness of the software at each stage and between each stage of the development life cycle." [RBC82]
- Validation
  Did we build the product right?
  Validation is the "determination of the correctness of the final program or software produced from a development project with respect to the user needs and requirements. Validation is usually accomplished by verifying each stage of the software development life cycle." [RBC82]

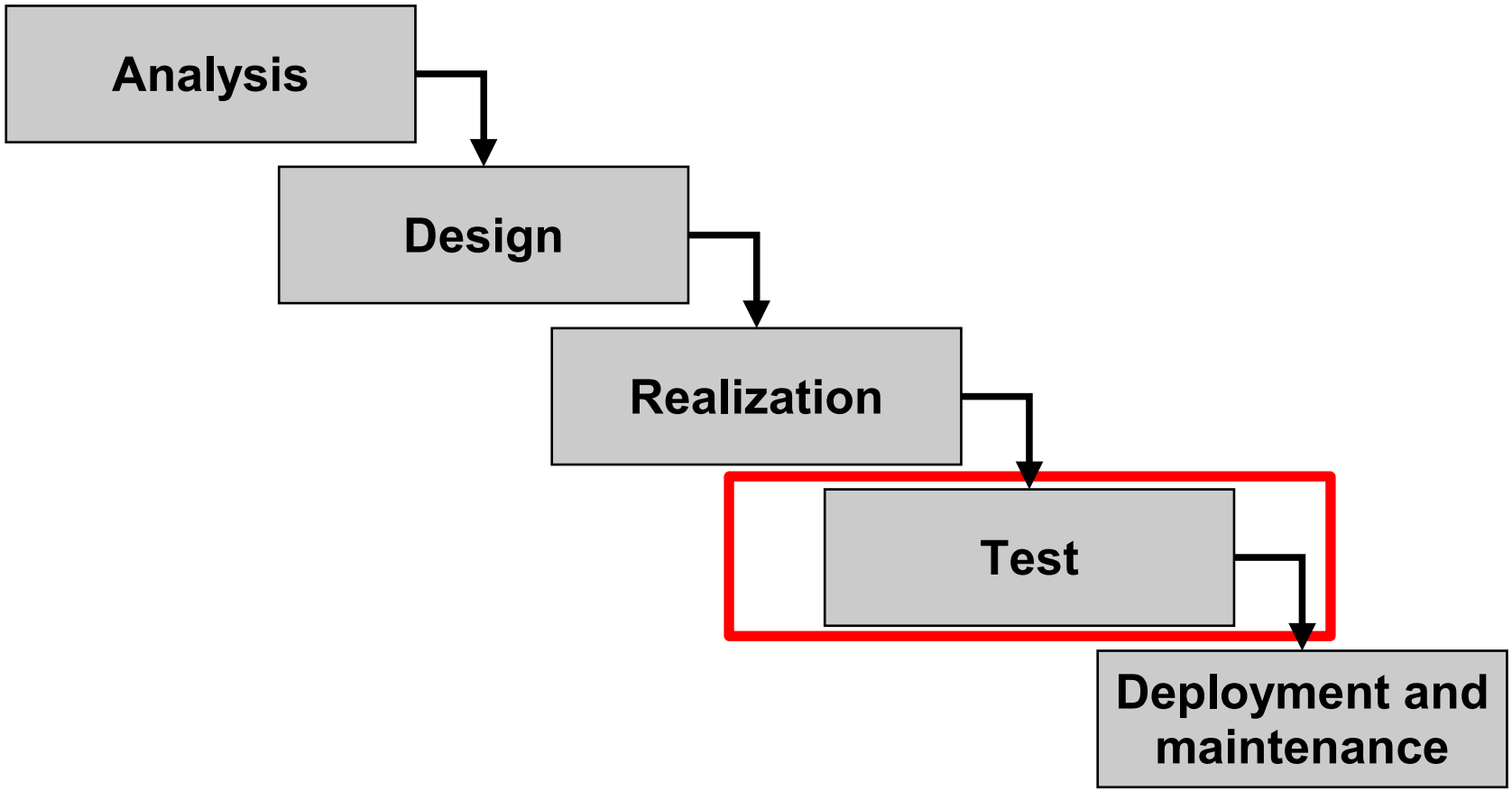
Definitions
Commercial Off-The-Shelf (COTS)

- Commercial Off-The-Shelf (COTS) software [ISTQB-GWP12]

  A software product that is
  - developed for the general market, i.e. for a large number of customers,
  - delivered to many customers in identical format.

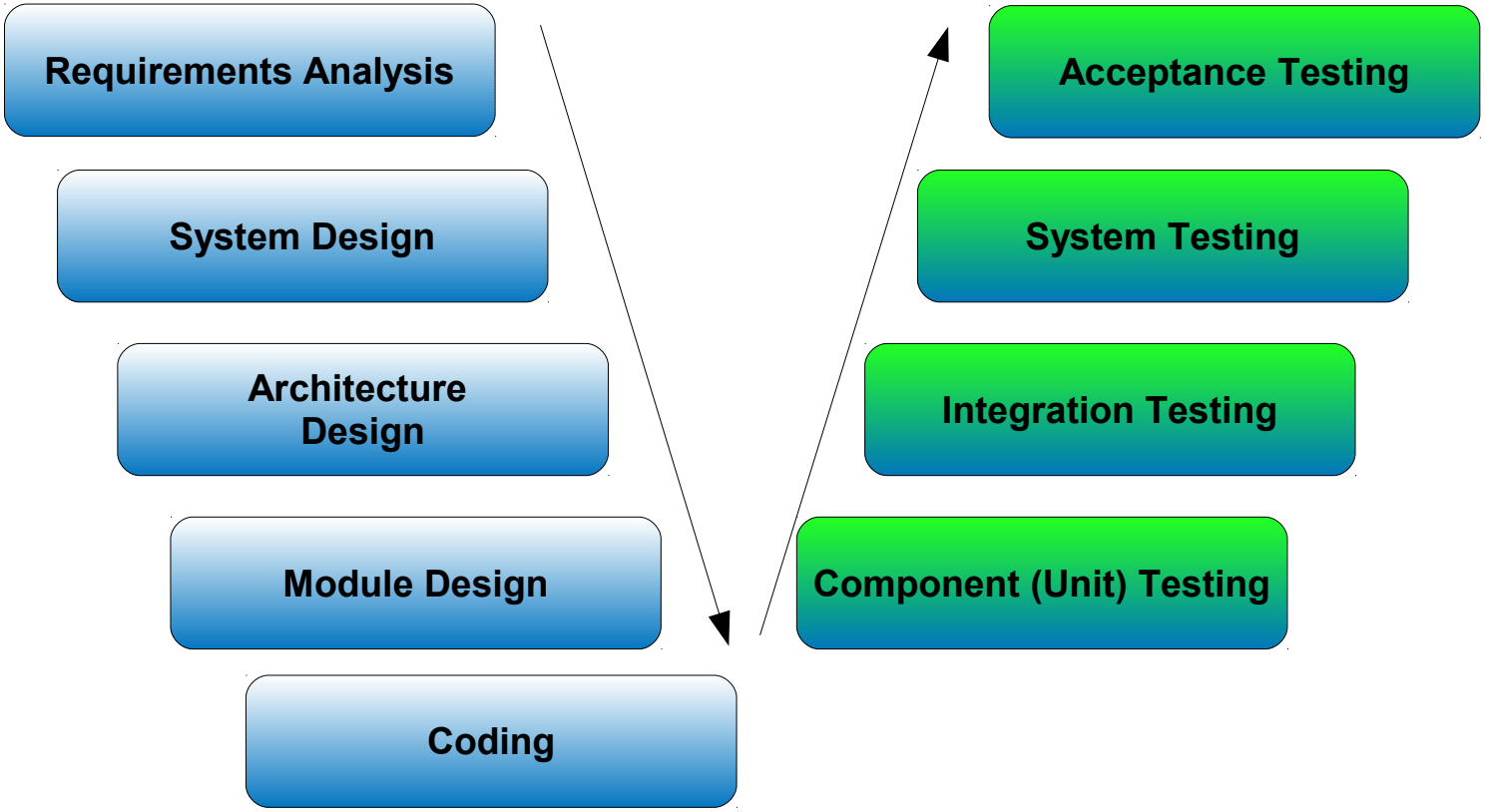
Software Development Models
Sequential Development Model

- Waterfall model



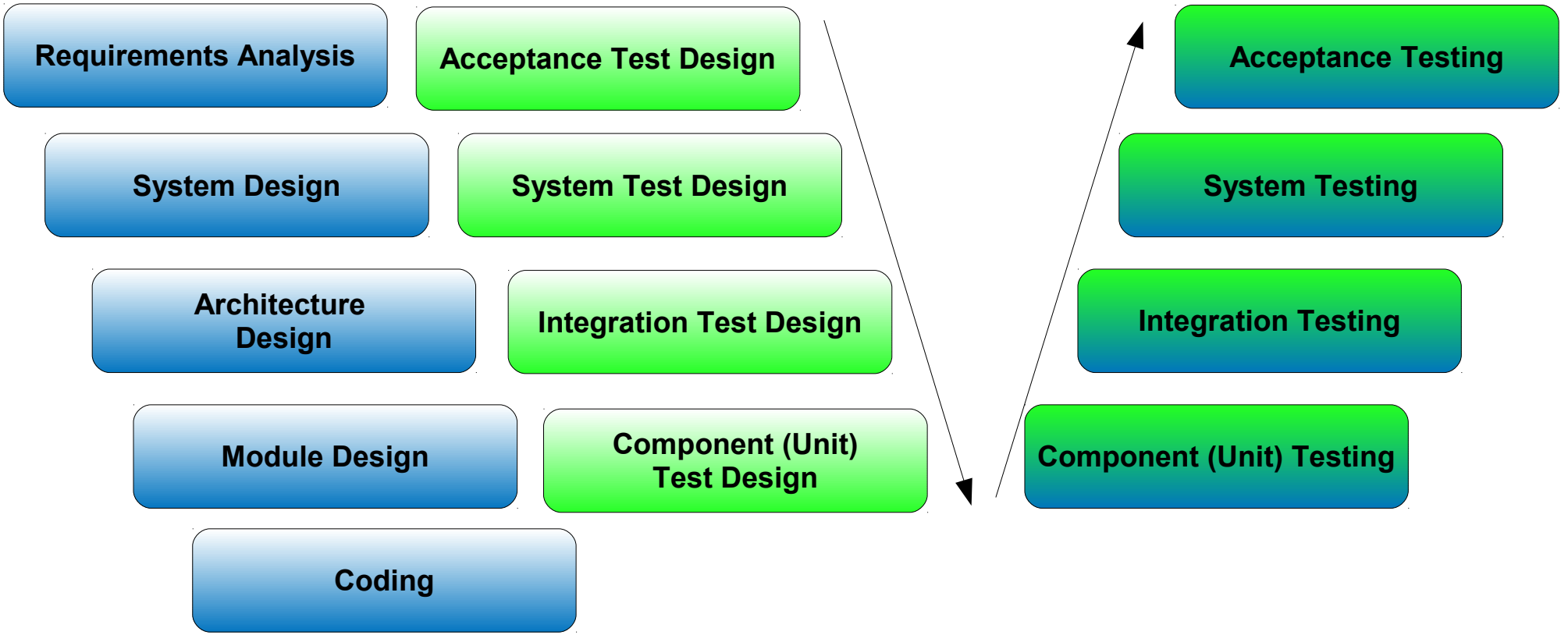
Software Development Models
Sequential Development Model

- V-Model following ISTQB



Software Development Models
Sequential Development Model

- V-Model following ISTQB – including test preparation activities



Software Development Models
Iterative-incremental Development Models

- Rational Unified Process

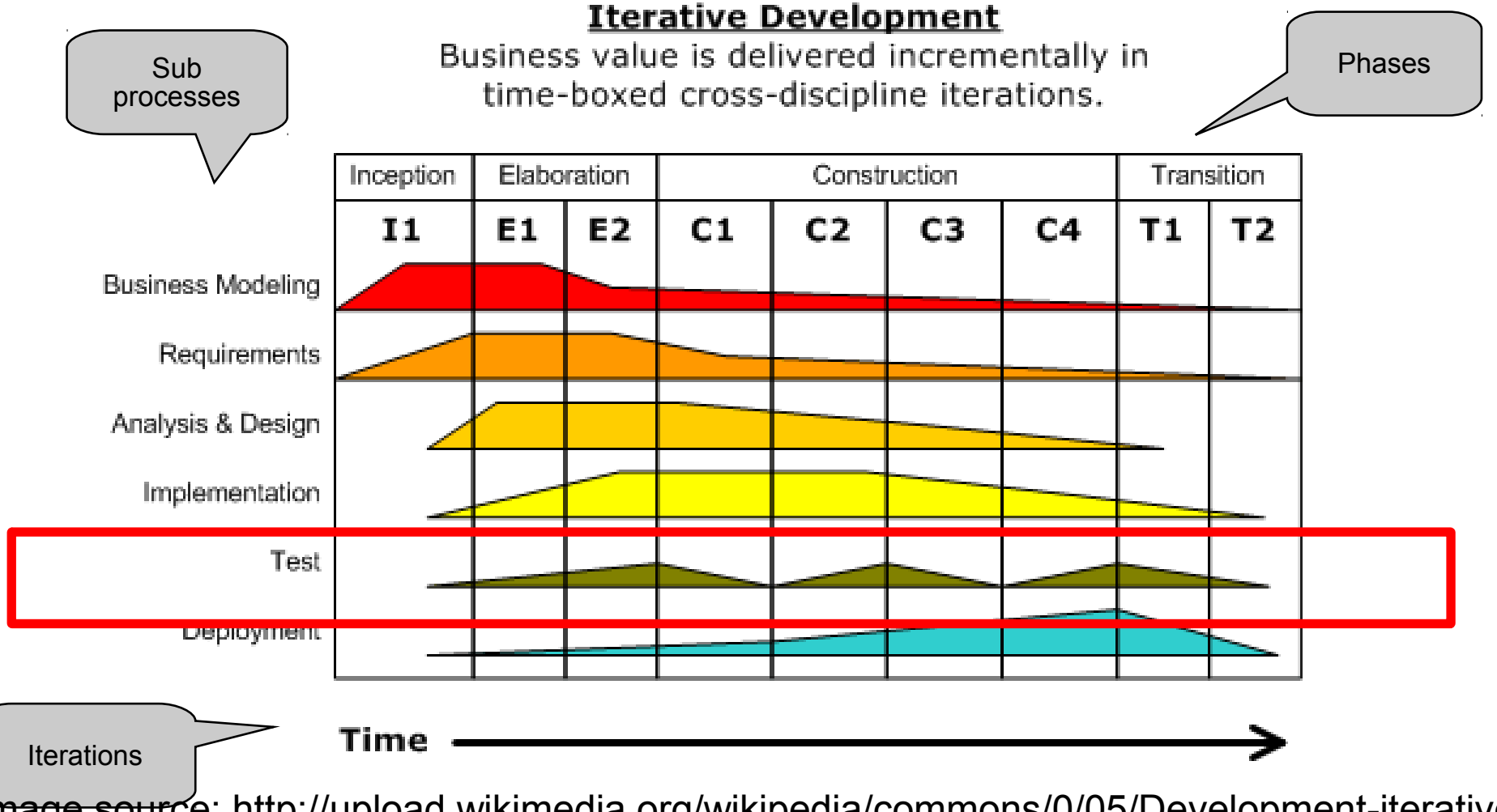


Image source: http://upload.wikimedia.org/wikipedia/commons/0/05/Development-iterative.gif

Software Development Models
Iterative-incremental Development Models

- Agile Software Development
- Agile manifesto [BBB+01]
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

Testing in Software Life Cycles
Iterative-incremental Development Models

Image source:
http://en.wikipedia.org/wiki/File:Agile_Software_Development_methodology.jpg

Testing in Software Life Cycles
Iterative-incremental Development Models

- Agile Software Development => Agile testing
- Agile testing involves testing from the customer point of view as early as possible – depending on availability and stability of code.
- Test automation plays a central role.
  Typical test execution proceeding after delivery:
  1. (Automated) smoke test / sanity check
  2. Execution of automated regression test suite
  3. Execution of manual tests concerning new implemented user stories / bug fixes
  4. Extending automated test suite

Testing in Software Life Cycles
Iterative-incremental Development Models

- Scrum

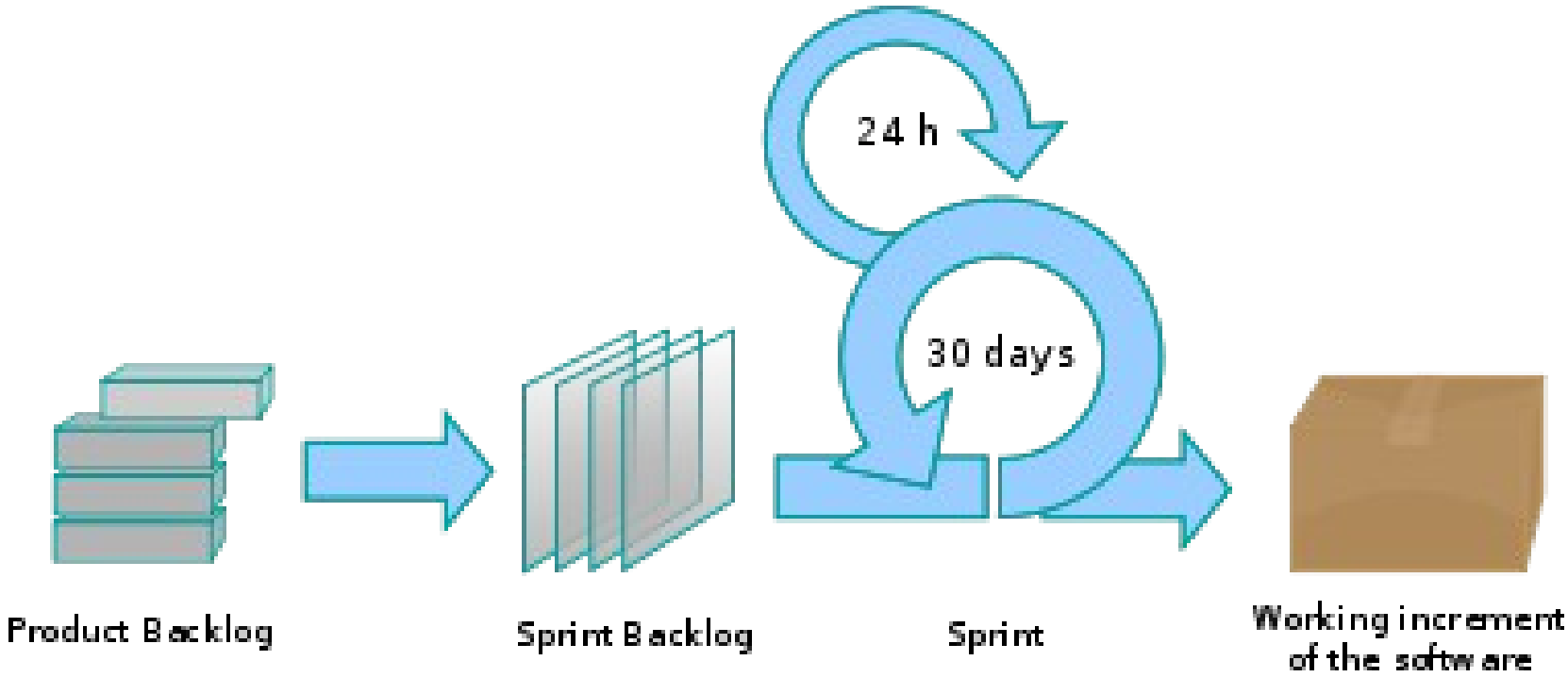


Image source: http://en.wikipedia.org/wiki/File:Scrum_process.svg

Testing in Software Life Cycles
Testing within a Life Cycle Model

- Test levels in software life cycle models
  - Development activity ⇔ Testing activity
  - Each test level has specific test objectives
  - The analysis and design of tests for a given test level should begin during the corresponding development activity
  - Testers should be involved in reviewing documents as soon as drafts are available in the development life cycle
  - Test levels can be combined or reorganized depending on the project or the system

Testing in Software Life Cycles
Testing within a Life Cycle Model

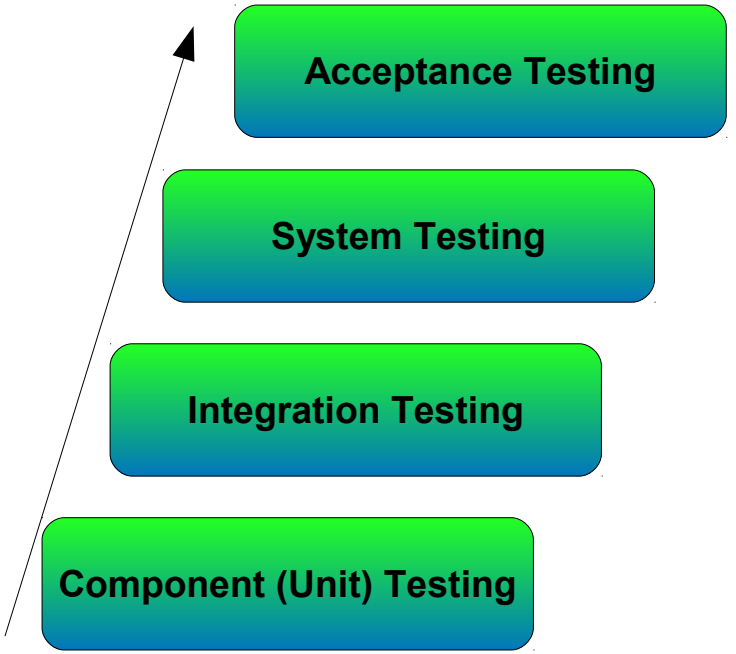
- Example: Integration of a Commercial Off-The-Shelf (COTS) software product into a system

  Purchaser may perform

  - Integration testing at the system level, e.g. integration to the infrastructure
  - Acceptance testing
    - Functional testing
    - Non-functional testing
    - User testing
    - Operational testing

Test Levels

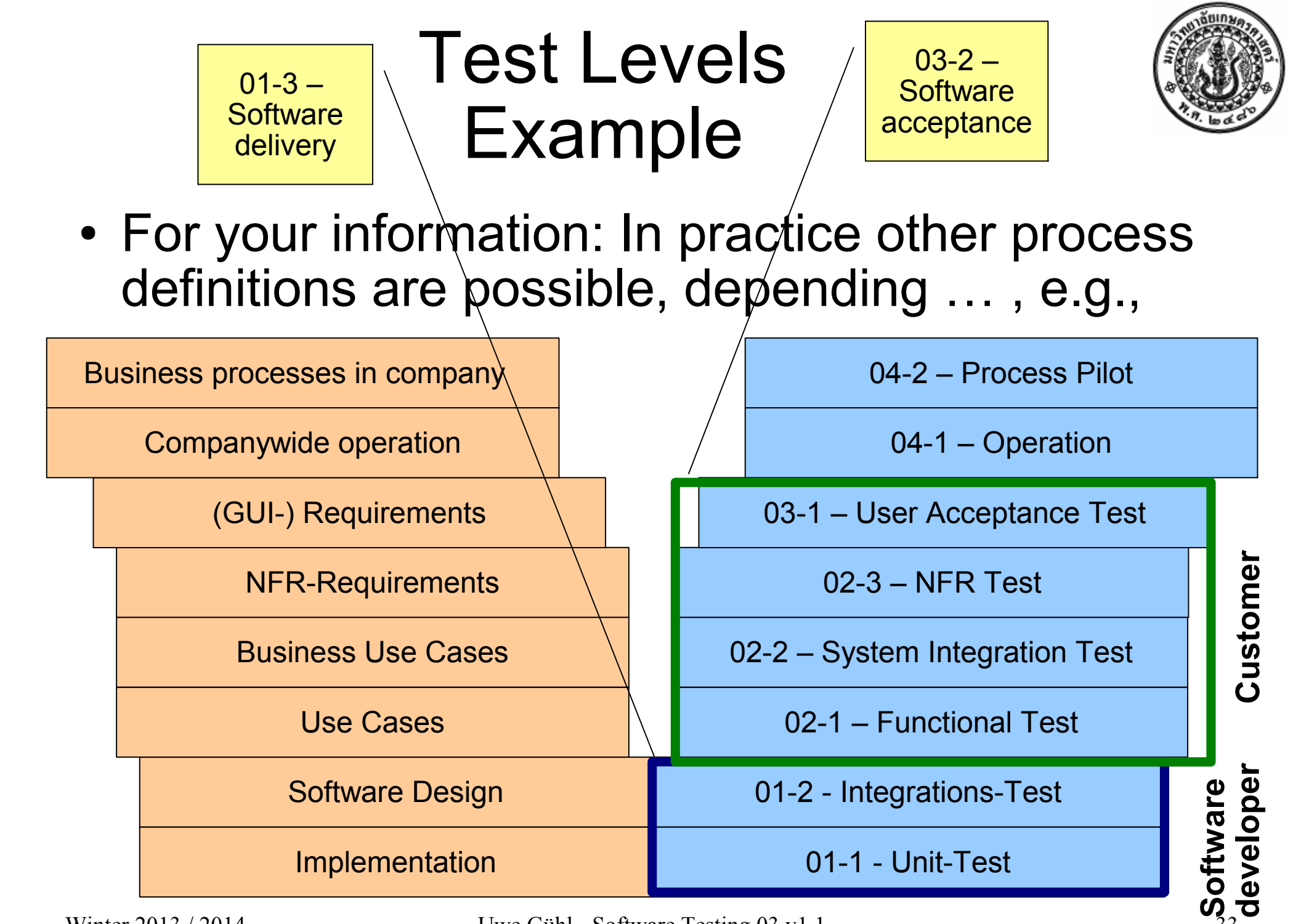
… following ISTQB



Test Levels

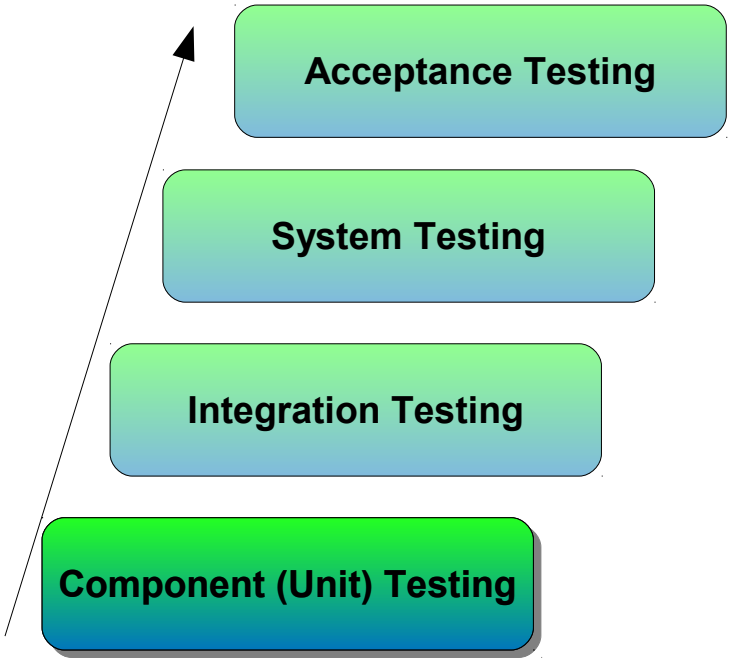
- For each test level could be identified:
  - the generic objectives,
  - the work product(s) being referenced for deriving test cases (i.e., the test basis),
  - the test object (i.e., what is being tested),
  - typical defects and failures to be found,
  - test harness requirements,
  - tool support,
  - specific approaches,
  - responsibilities.

Test Levels Example



- For your information: In practice other process definitions are possible, depending … , e.g.,

Test Levels
Component Testing



Test Levels
Component Testing

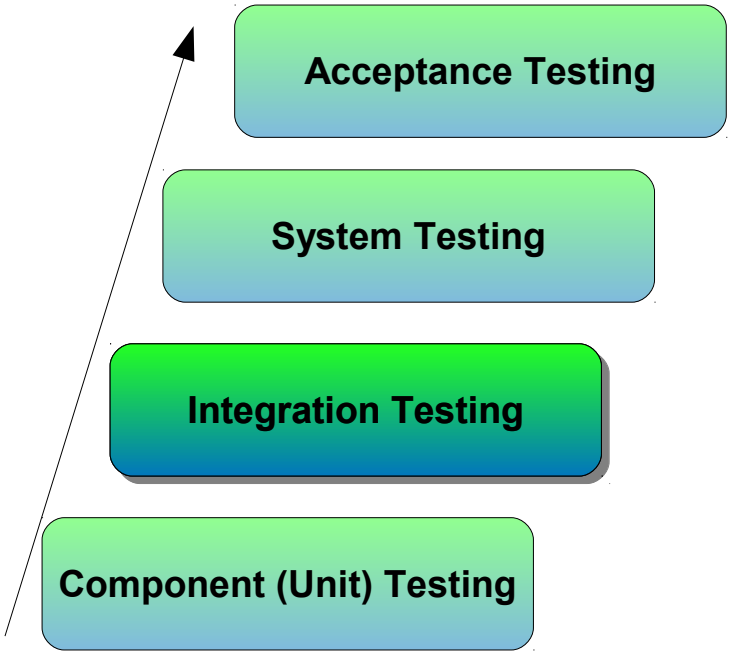
- Also known as unit, module or program testing
- Component testing searches for defects in, and verifies the functioning of,
  - software modules,
  - programs,
  - objects,
  - classes, etc.,

  that are separately testable.
- Test-driven development: Prepare and automate test cases before coding.

Test Levels
Component Testing

- Test basis
  - Component requirements,
  - Detailed design,
  - Code.
- Typical test objects
  - Components,
  - Programs,
  - Data conversion / migration programs,
  - Database modules.

Test Levels
Integration Testing



Test Levels
Integration Testing

- Integration testing tests
  - interfaces between components;
  - interactions with different parts of a system, such as the operating system, file system and hardware;
  - interfaces between systems.

Test Levels
Integration Testing

- Test basis
  - Software and system design,
  - Architecture,
  - Workflows,
  - Use cases.
- Typical test objects
  - Subsystems,
  - Database implementation,
  - Infrastructure,
  - Interfaces,
  - System configuration and configuration data.

Test Levels
Integration Testing

It is important to distinguish:

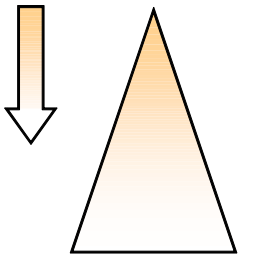
- Component integration testing
  tests the interactions between software components and is done after component testing
- System integration testing
  tests the interactions between different systems or between hardware and software and may be done after system testing.
  - In this case, the developing organization may control only one side of the interface. Could be a risk.
  - Business processes implemented as workflows may involve a series of systems.
  - Cross-platform issues may be significant.

Test Levels
Integration Testing

- The greater the scope of integration, the more difficult it becomes to isolate defects to a specific component or system. This may lead to
  - increased risk,
  - additional time for troubleshooting.
- Ideally, testers should understand the architecture and influence integration planning.

Test Levels
Integration Testing

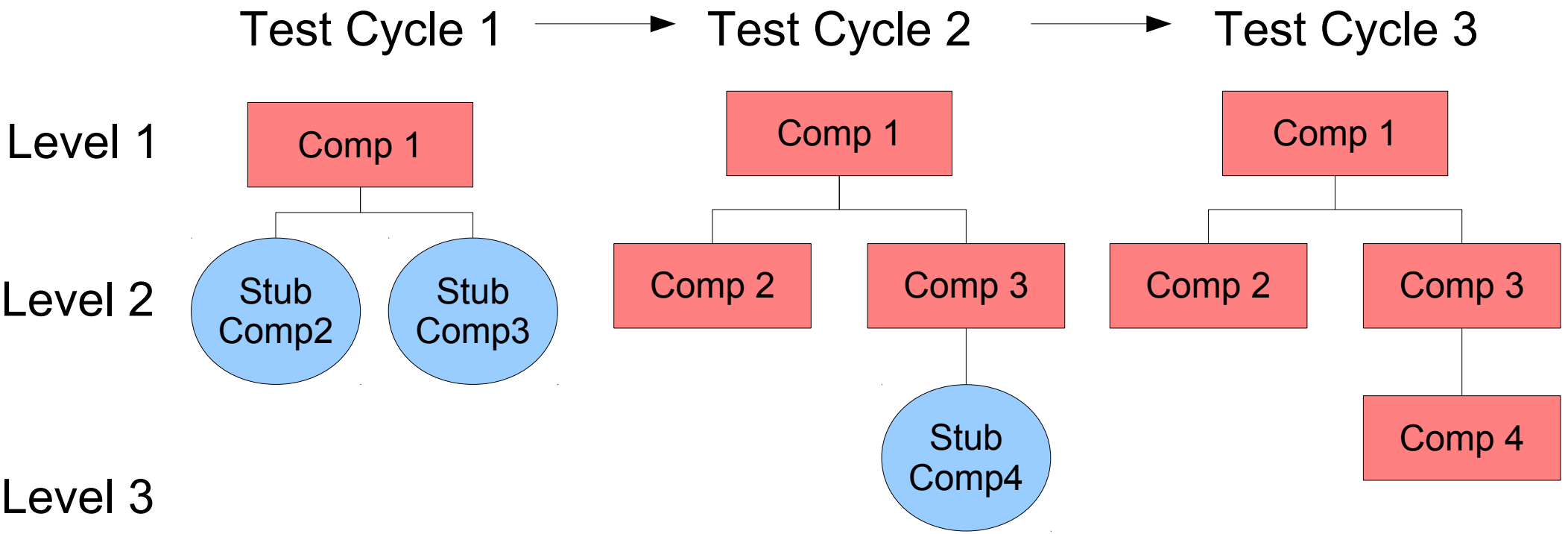
- **Top-down testing** approach to integration testing
  - The component at the top of the component hierarchy is tested first, lower level components are simulated by stubs.
    - A stub is a skeletal or special-purpose implementation of a software component, used to develop or test a component that calls or is otherwise dependent on it. It replaces a called component. [After IEEE 610]
  - Tested components are then used to test lower level components.
  - The process is repeated until the lowest level components have been tested.



Test Levels
Integration Testing

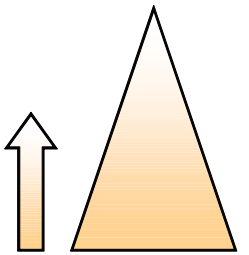
- **Top-down testing** approach to integration testing

  Stubs required, less drivers



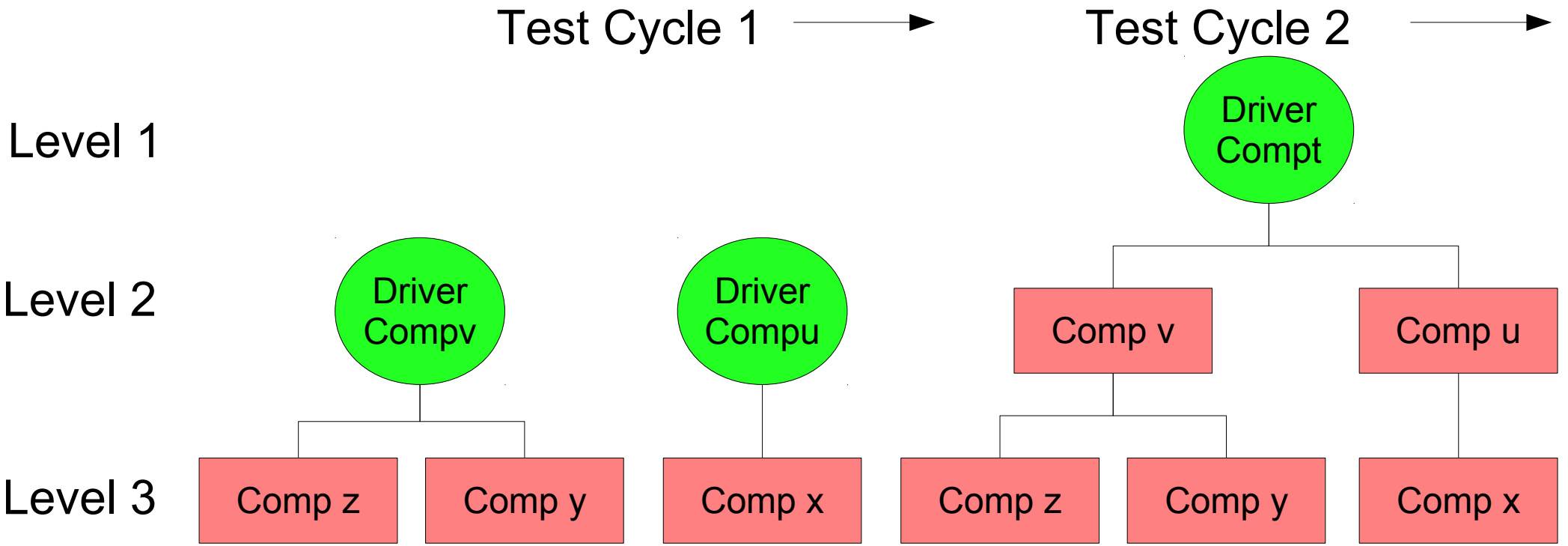
Test Levels
Integration Testing

- **Bottom-up testing** approach to integration testing
  - The lowest level components are tested first, and then used to facilitate the testing of higher level components.
  - This process is repeated until the component at the top of the hierarchy is tested.

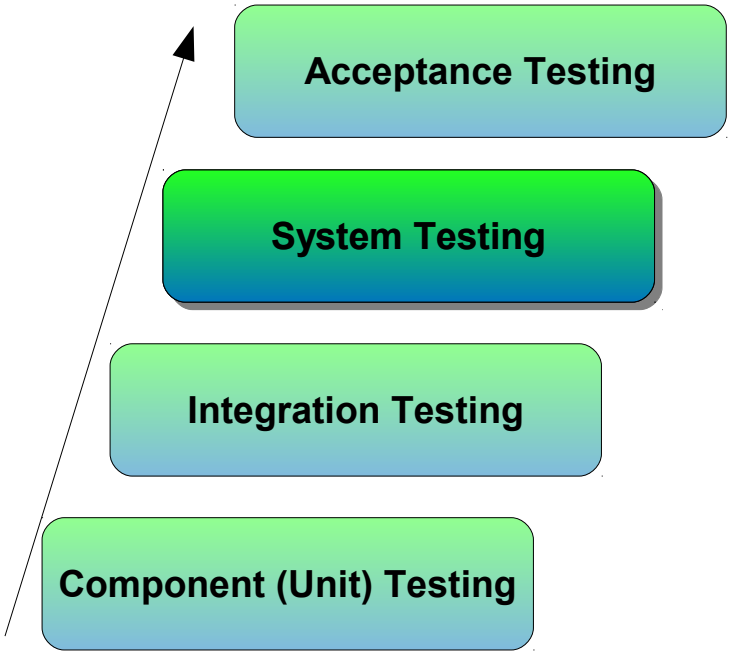


Test Levels
Integration Testing

- **Bottom-up testing** approach to integration testing
  Drivers required, less stubs



Test Levels
System Testing



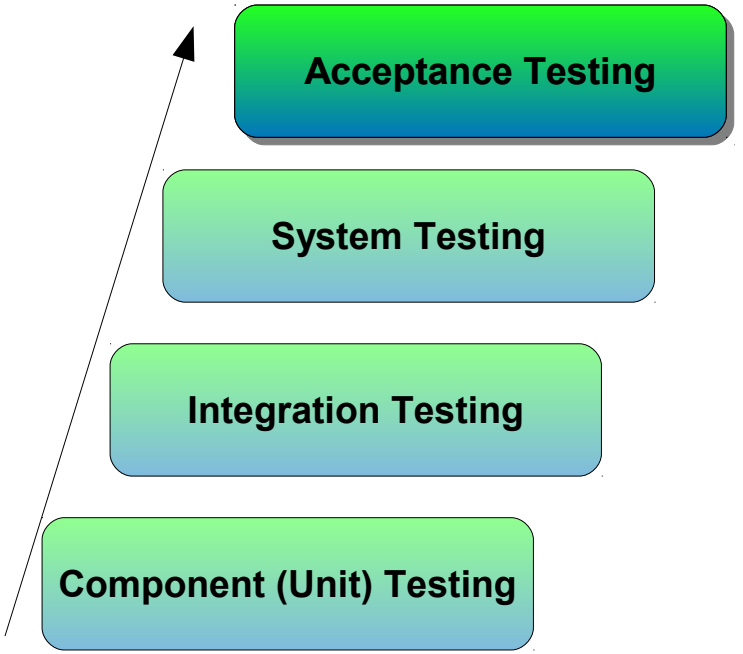
Test Levels
System Testing

- System testing is concerned with the behaviour of a whole system/product.
- The test environment should correspond to the final target or production environment as much as possible to minimize the risk of environment-specific failures.
- System testing should investigate functional and non-functional requirements of the system, and data quality characteristics.

Test Levels
System Testing

- Test basis
  - System and software requirement specification
  - Use cases
  - Functional specification
  - Risk analysis reports
- Typical test objects
  - System, user and operation manuals
  - System configuration and configuration data

Test Levels
Acceptance Testing



Test Levels
Acceptance Testing

- Acceptance testing should establish confidence
  - in the system,
  - parts of the system, or
  - specific non-functional characteristics of the system.
- Acceptance testing often assesses the system's readiness for deployment and use.
- Finding defects is not the main focus.
- Acceptance testing is often the responsibility of the customers or users of a system, other stakeholders could be involved.

Test Levels
Acceptance Testing

Typical forms of acceptance testing (1/3):

- User acceptance testing
  - Typically verifies the fitness for use of the system by business users.
- Operational (acceptance) testing
  - The acceptance of the system by the system administrators, including for example:
    - Testing of backup/restore
    - Disaster recovery
    - User management
    - Data load and migration tasks
    - Periodic checks of security vulnerabilities

Test Levels
Acceptance Testing

Typical forms of acceptance testing (2/3):

- Contract acceptance testing
  is performed against a contract's acceptance criteria for producing custom-developed software.
- Regulation acceptance testing
  is performed against any regulations that must be adhered to, such as government, legal or safety regulations.

Test Levels
Acceptance Testing

Typical forms of acceptance testing (3/3):

- Alpha testing
  is performed at the developing organization's site but not by the developing team.
- Beta testing, or field-testing
  is performed by customers or potential customers at their own locations.

… both, Alpha and Beta testing, are done to get feedback from potential or existing customers in their market before the software product is put up for sale commercially

Test Levels
Acceptance Testing

- Test basis
  - User requirements
  - System requirements
  - Use cases
  - Business processes
  - Risk analysis reports
- Typical test objects
  - Business processes on fully integrated system
  - Operational and maintenance processes
  - User procedures
  - Forms
  - Reports
  - Configuration data

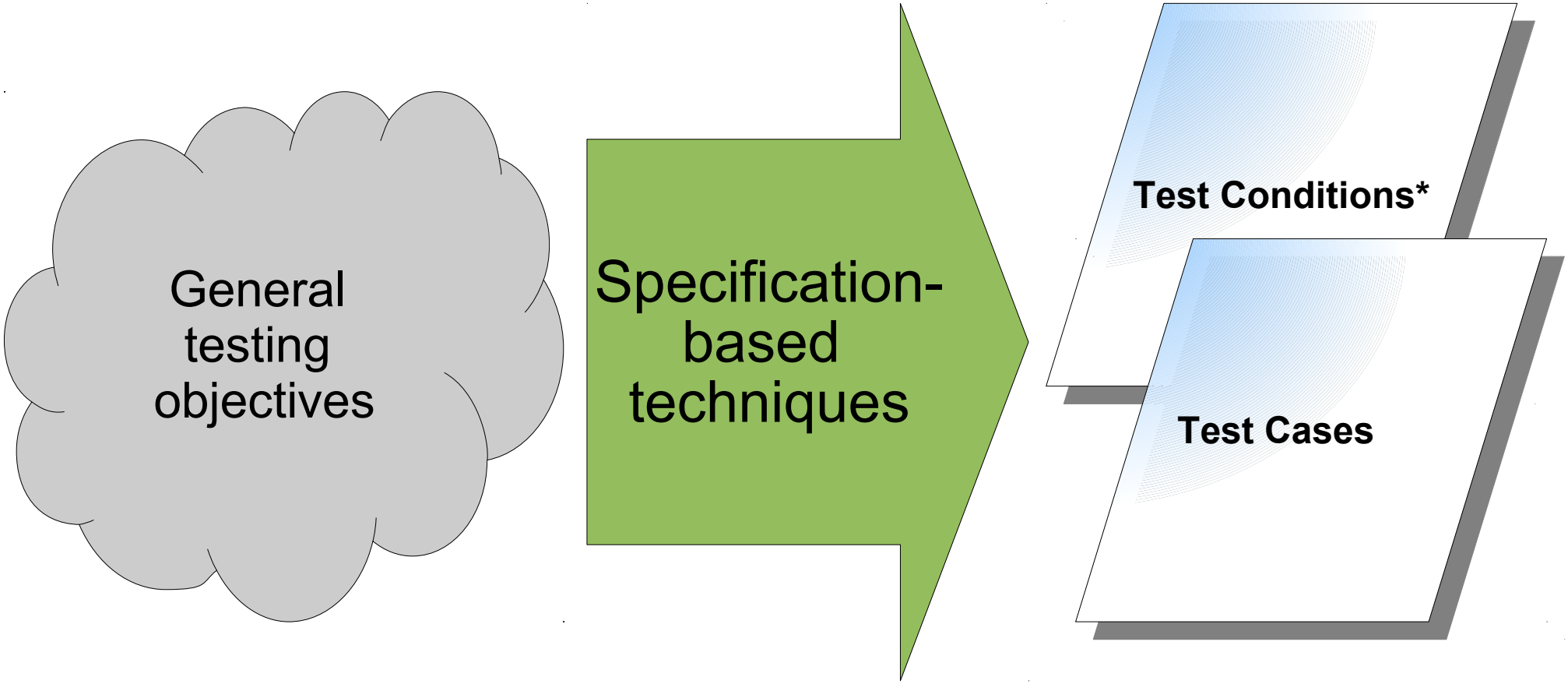
Test Types

- A test type is focused on a particular test objective, which could be:
  - A function to be performed by the software.
  - A non-functional quality characteristic, such as reliability or usability.
  - The structure or architecture of the software.
  - Change related, for example
    - confirming that defects have been fixed (re-testing or confirmation testing),
    - looking for unintended changes (regression testing).

Test Types
Functional testing

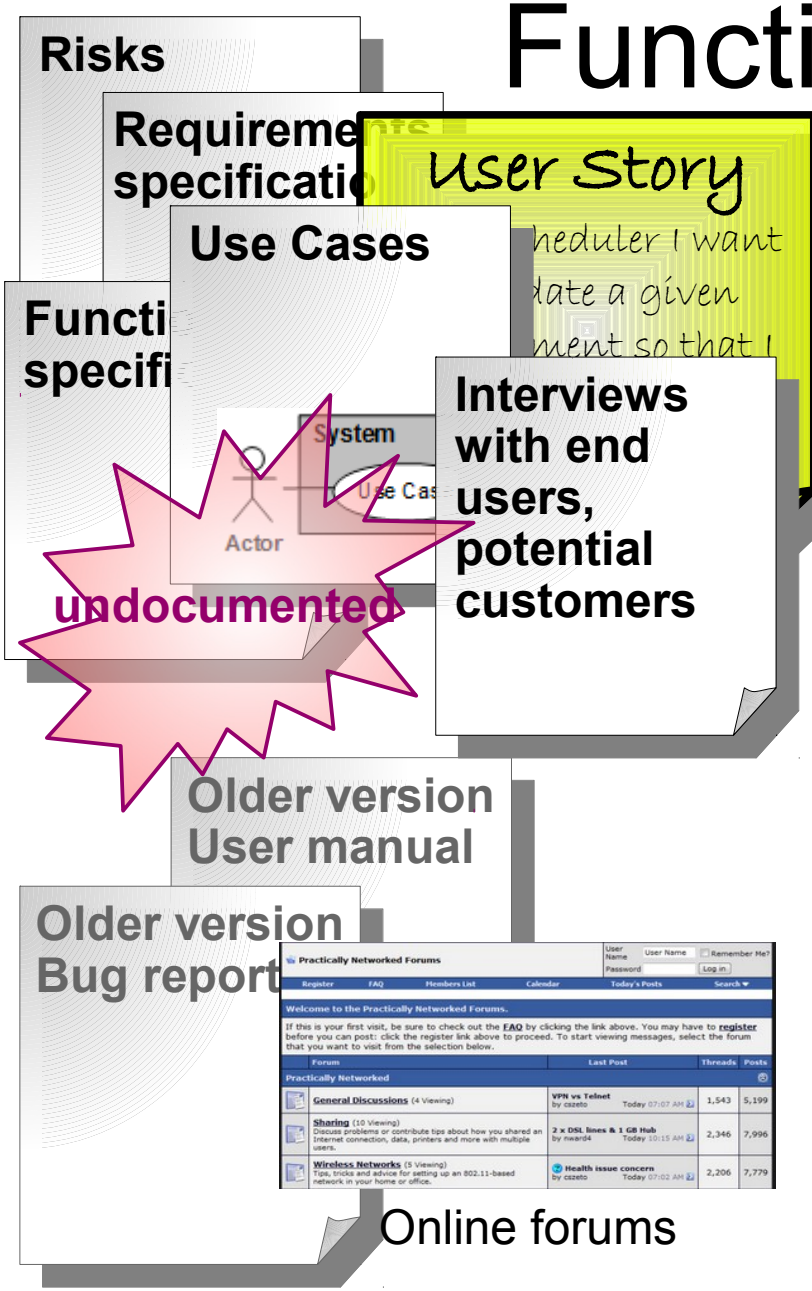
- Functional testing to test "what" the system does.
- Functional testing considers the external behaviour of the software (black-box testing).
- Functional tests are based on
  - functions
  - features and
  - their interoperability with specific systems
- Functional tests may be performed at all test levels (e.g., tests for components may be based on a component specification).

Test Types
Functional testing

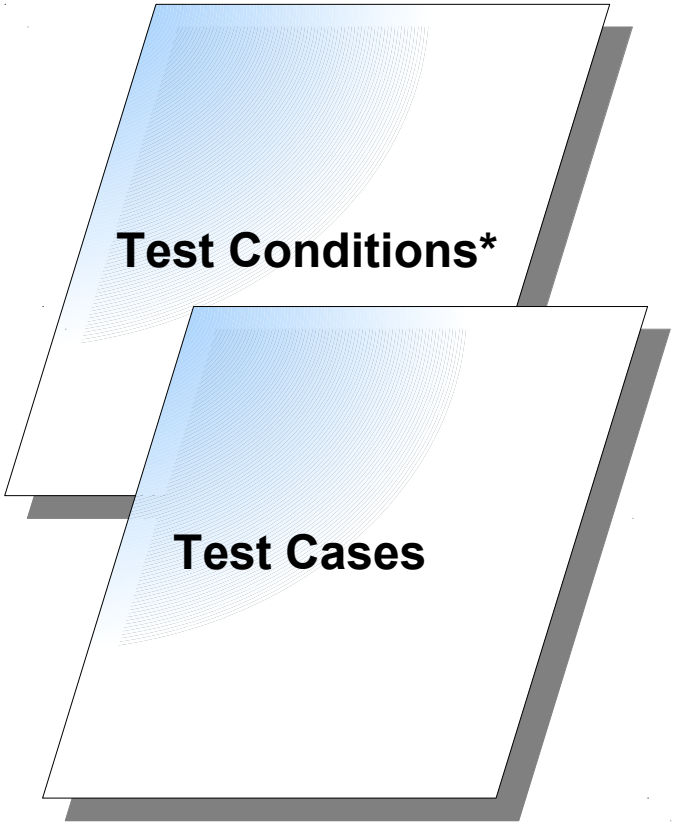


* Test condition = An item or event of a component or system that could be verified by one or more test cases, e. g. a function, transaction, feature, quality attribute, or structural element [ISTQB-GWP12].

Test Types
Functional testing



Specification-based techniques



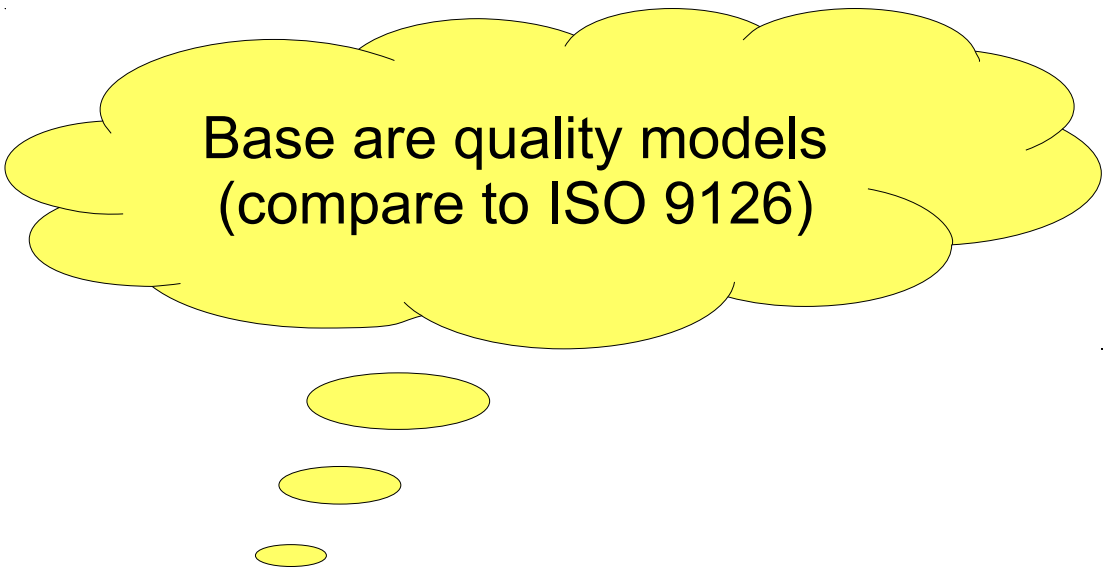
* Test condition = An item or event of a component or system that could be verified by one or more test cases, e. g. a function, transaction, feature, quality attribute, or structural element [ISTQB-GWP12].

Test Types
Functional testing

- Types of functional testing
  - Security testing
    investigates the functions (e.g., a firewall) relating to detection of threats, such as viruses, from malicious outsiders.
  - Interoperability testing
    evaluates the capability of the software product to interact with one or more specified components or systems.

Test Types
Non-functional testing

- Non-functional testing to test "How" the system does.
- Non-functional testing to measure characteristics of systems and software that can be quantified, such as response times for performance testing
- Typical non-functional tests:
  - performance testing,
  - load testing,
  - stress testing,
  - usability testing,
  - maintainability testing,
  - reliability testing, and
  - portability testing



Test Types
Structural Testing

- Testing of Software Structure / Architecture
- Structural tests may be performed at all test levels, for example in
  - Component testing and component integration testing

    A software model could be used for structural testing, e.g.,
    - a control flow model
    - a menu structure model
  - System, system integration or acceptance testing

    A business model could be used as well, e.g.
    - business use cases

Test Types
Structural Testing

- Coverage
  is the extent that a structure has been exercised by a test suite, expressed as a percentage of the items being covered.
- Tools measure the code coverage of elements, such as
  - Statements
  - Decisions

Test Types
Re-testing and Regression Testing

- Testing related to changes
- Re-testing
  - After a defect is detected and fixed, the software should be re-tested to confirm that the original defect has been successfully removed.
  - This is called confirmation.
- Regression Testing
  - Repeated testing of an already tested program, after modification, to discover any defects introduced or uncovered as a result of the change(s).
  - These extent is based on the risk of not finding defects in software that was working previously.

Maintenance Testing

- Background
  Once deployed, a software system is often in service for years or decades.
- Maintenance testing is done on an existing operational system, and is triggered by
  - modifications,
  - migration, or
  - retirement of the software or system.

Maintenance Testing

- Modifications should be planned and may include
  - enhancement changes (e.g., release-based),
  - emergency changes,
  - changes of environment, such as operating system or database upgrades,
  - upgrade of Commercial-Off-The-Shelf (COTS) software,
  - patches to correct newly exposed or discovered vulnerabilities of the operating system.

Maintenance Testing

- Maintenance testing concerning what has changed.
  Examples
  - Change from one platform to another
    Proposal: Operational tests of the new environment as well as of the changed software.
  - Data migration from another application into the system being maintained
    Proposal: Database tests, system tests

Maintenance Testing

- Maintenance testing concerning
  what has not be changed ⇨ Regression testing
  - Scope of regression testing is related to
    - risk of the change,
    - size of the existing system,
    - size of the change.
  - Impact analysis to
    - determine how the existing system may be affected by changes and
    - decide how much regression testing to do.
    - determine the regression test suite.

Maintenance Testing

- Retirement of a system
  ⇨Maintenance testing may include the testing of
  - data migration
  - archiving if long data-retention periods are required.

- Challenge
  Maintenance testing can be difficult if
  - specifications are out of date or missing,
  - testers with domain knowledge are not available.

Sources

- [BBB+01] Beck, Beedle, van Bennekum, et al.: Manifesto for agile Software Development, 2001, http://agilemanifesto.org/
- [IEEE 610] IEEE 610.12:1990. Standard Glossary of Software Engineering Terminology.
- [ISTQB-CTFLS11] International Software Testing Qualifications Board: Certified Tester Foundation Level Syllabus, Released Version 2011, http://www.istqb.org/downloads/syllabi/foundation-level-syllabus.html
- [ISTQB-GWP12] Glossary Working Party of International Software Testing Qualifications Board: Standard glossary of terms used in Software Testing, Version 2.2, 2012, http://www.istqb.org/downloads/glossary.html
- [RBC82] Adrion, W. Richards, Martha A. Branstad, and John C. Cherniavsky. "Validation, Verification, and Testing of Computer Software," Computing Surveys, June 1982, pp. 159-192
- [Wik14] Wikipedia: ISO/IEC 9126, 2014, http://en.wikipedia.org/wiki/ISO/IEC_9126