1. Software development models
- *1.1 Sequential model*
- *1.2 Iterative-incremental model*
- *1.3 Testing within a life-cycle model*

2. Test Levels
- *2.1 Component testing*
- *2.2 Integration testing*
- *2.3 System testing*
- *2.4 Acceptance testing*

3. Test Types
- *3.1 Testing of function*
- *3.2 Testing of non-functional software characteristics*
- *3.3 Testing of software structure*
- *3.4 Testing related to changes*

4. Maintenance testing

IN 3240/IN4240 Software testing

Testing throughout the software life cycle

1. Software development models
2. Test Levels
3. Test Types
4. Maintenance testing

1. Software development models
- *1.1 Sequential model*
- *1.2 Iterative-incremental model*
- *1.3 Testing within a life-cycle model*

2. Test Levels
- *2.1 Component testing*
- *2.2 Integration testing*
- *2.3 System testing*
- *2.4 Acceptance testing*

3. Test Types
- *3.1 Testing of function*
- *3.2 Testing of non-functional software characteristics*
- *3.3 Testing of software structure*
- *3.4 Testing related to changes*

4. Maintenance testing

Software development models

- Testing is important in the software development life cycle
- The life cycle model will determine how to organize the testing
- Testing is highly related to software development activities

Waterfall model

1. Software development models
- *1.1 Sequential model*
- *1.2 Iterative-incremental model*
- *1.3 Testing within a life-cycle model*

2. Test Levels
- *2.1 Component testing*
- *2.2 Integration testing*
- *2.3 System testing*
- *2.4 Acceptance testing*

3. Test Types
- *3.1 Testing of function*
- *3.2 Testing of non-functional software characteristics*
- *3.3 Testing of software structure*
- *3.4 Testing related to changes*

4. Maintenance testing

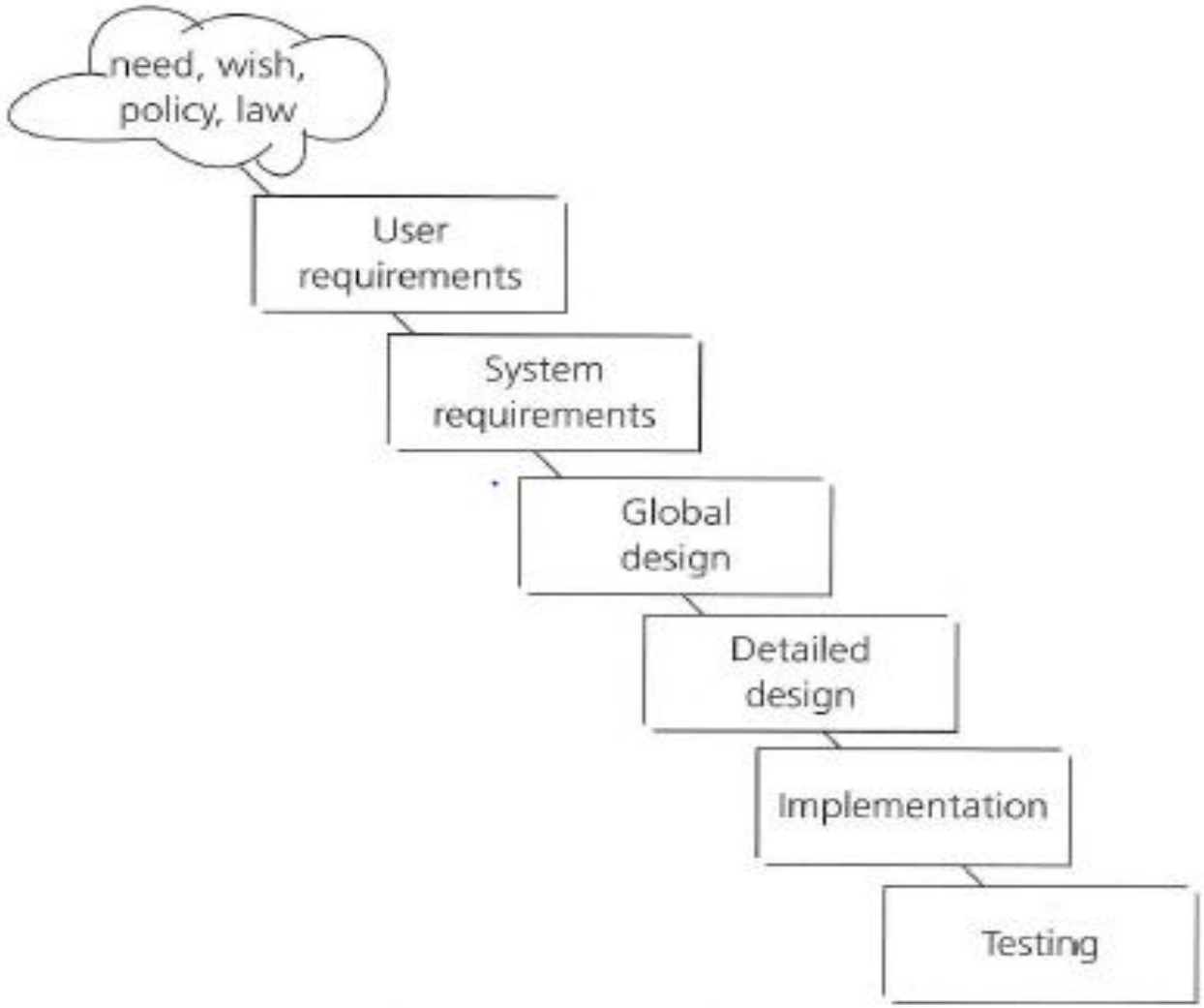


FIGURE 2.1 Waterfall model

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Iterative/incremental development model

Examples of iterative and incremental development models

- Agile
- Rational Unified Process (RUP)
- Scrum
- Kanban
- Spiral (and prototyping)

CI/CD and DevOps are practices that can be applied within iterative and incremental development models

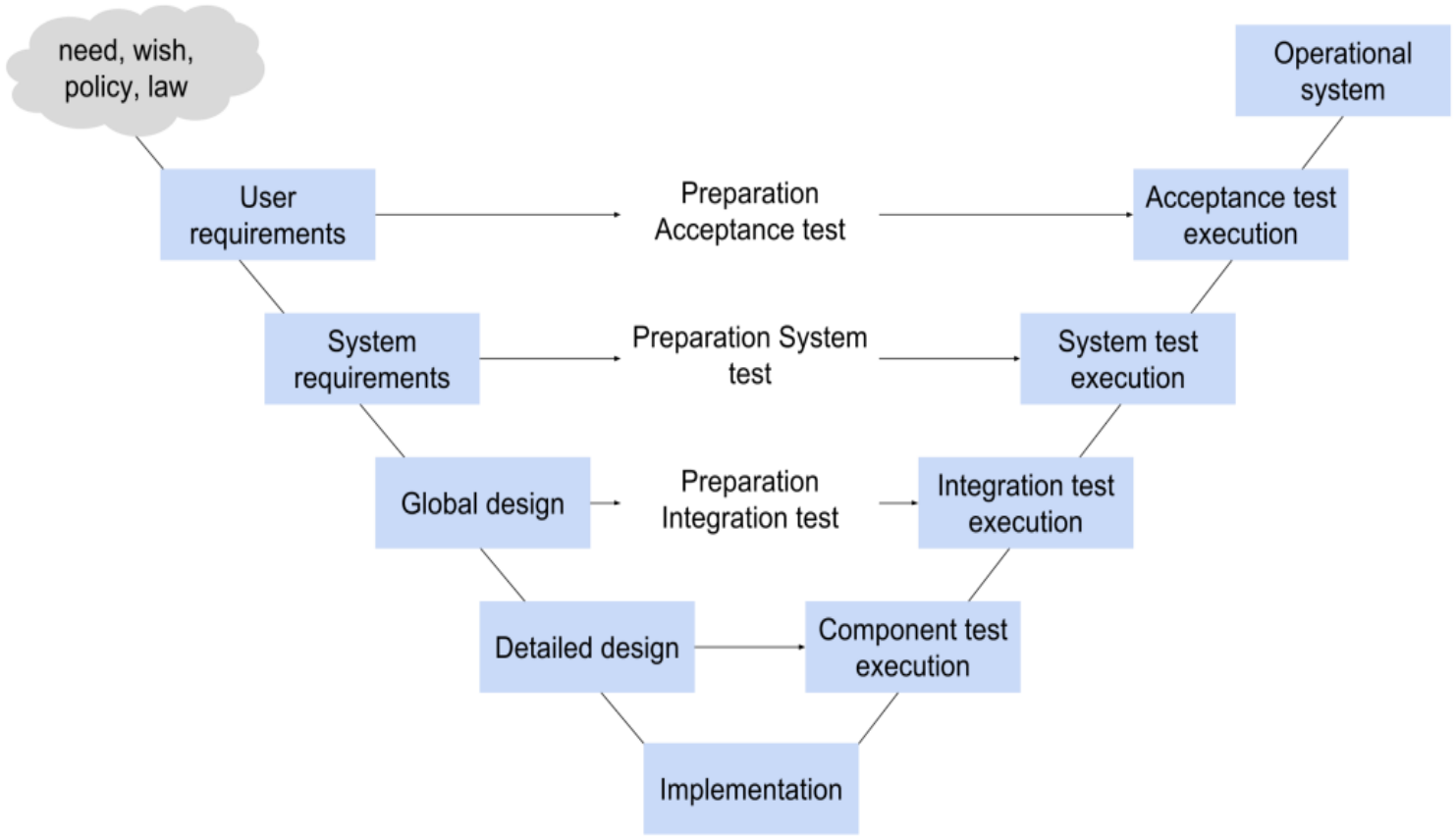
The V-model

1. *Software development models*
- *1.1 Sequential model*
- *1.2 Iterative-incremental model*
- *1.3 Testing within a life-cycle model*

2. *Test Levels*
- *2.1 Component testing*
- *2.2 Integration testing*
- *2.3 System testing*
- *2.4 Acceptance testing*

3. *Test Types*
- *3.1 Testing of function*
- *3.2 Testing of non-functional software characteristics*
- *3.3 Testing of software structure*
- *3.4 Testing related to changes*

4. *Maintenance testing*



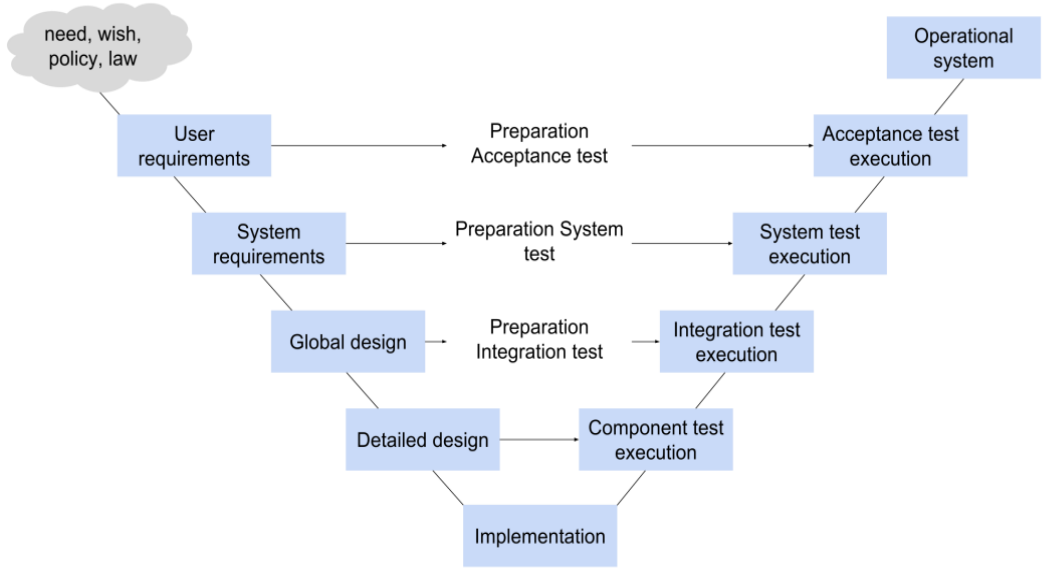
The V-model

1. Software development models
- *1.1 Sequential model*
- *1.2 Iterative-incremental model*
- *1.3 Testing within a life-cycle model*

2. Test Levels
- *2.1 Component testing*
- *2.2 Integration testing*
- *2.3 System testing*
- *2.4 Acceptance testing*

3. Test Types
- *3.1 Testing of function*
- *3.2 Testing of non-functional software characteristics*
- *3.3 Testing of software structure*
- *3.4 Testing related to changes*

4. Maintenance testing



- Testing needs to begin as early as possible in the life cycle.
- Testing can be integrated into each phase of the life cycle.
- Within the V-model, validation testing takes place especially during
  - the early stages, i.e. reviewing the user requirements
  - and late in the life cycle, i.e. during user acceptance testing

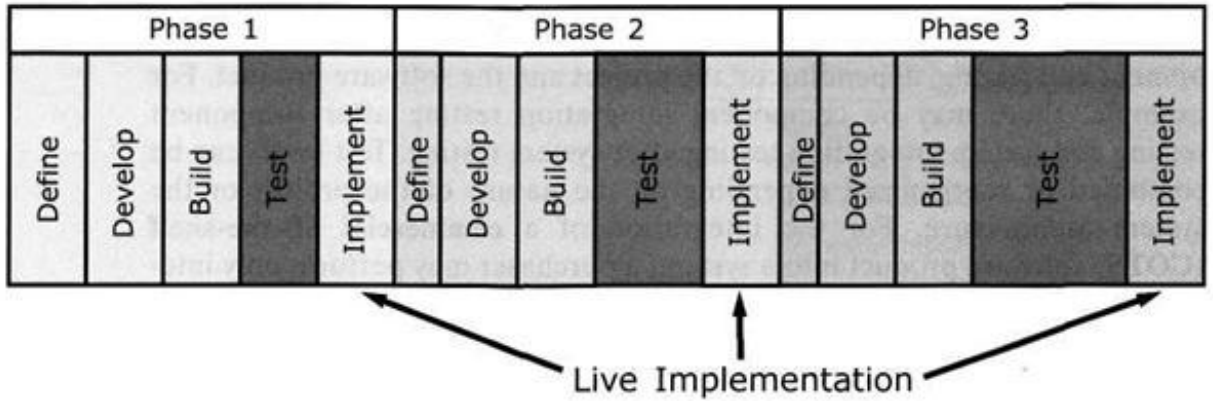
Iterative/incremental development model

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing



Iterative-incremental development is the process of establishing requirements, designing, building and testing a system carried out as a series of shorter development cycles.

An increment, added to others developed previously, forms a growing partial system, which should also be tested.

Regression testing is increasingly important to all iteration phases after the first one.

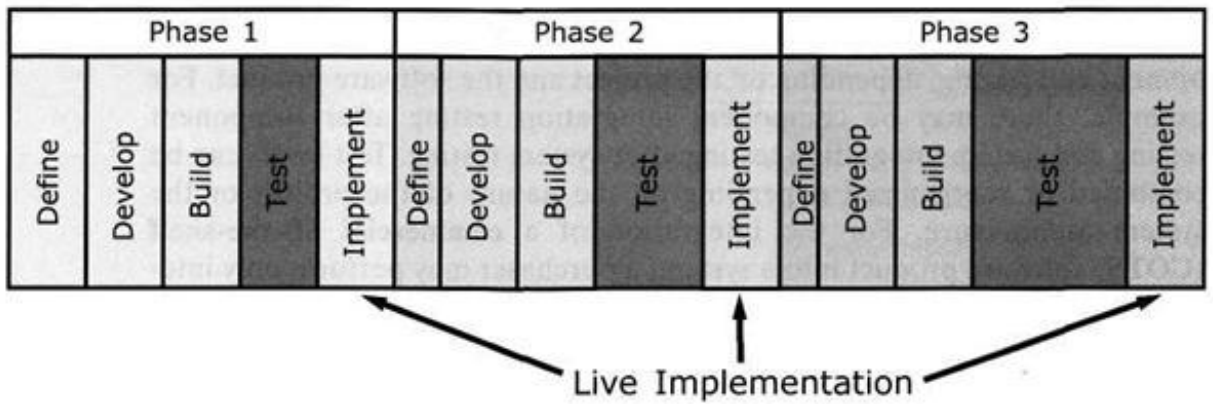
Iterative/incremental development model

1. Software development models
- *1.1 Sequential model*
- *1.2 Iterative-incremental model*
- *1.3 Testing within a life-cycle model*

2. Test Levels
- *2.1 Component testing*
- *2.2 Integration testing*
- *2.3 System testing*
- *2.4 Acceptance testing*

3. Test Types
- *3.1 Testing of function*
- *3.2 Testing of non-functional software characteristics*
- *3.3 Testing of software structure*
- *3.4 Testing related to changes*

4. Maintenance testing



The testing is often less formal

Common issues:

- More regression testing
- Defects outside the scope of the iteration or increment may be forgotten
- Less thorough testing

Testing within a life cycle model

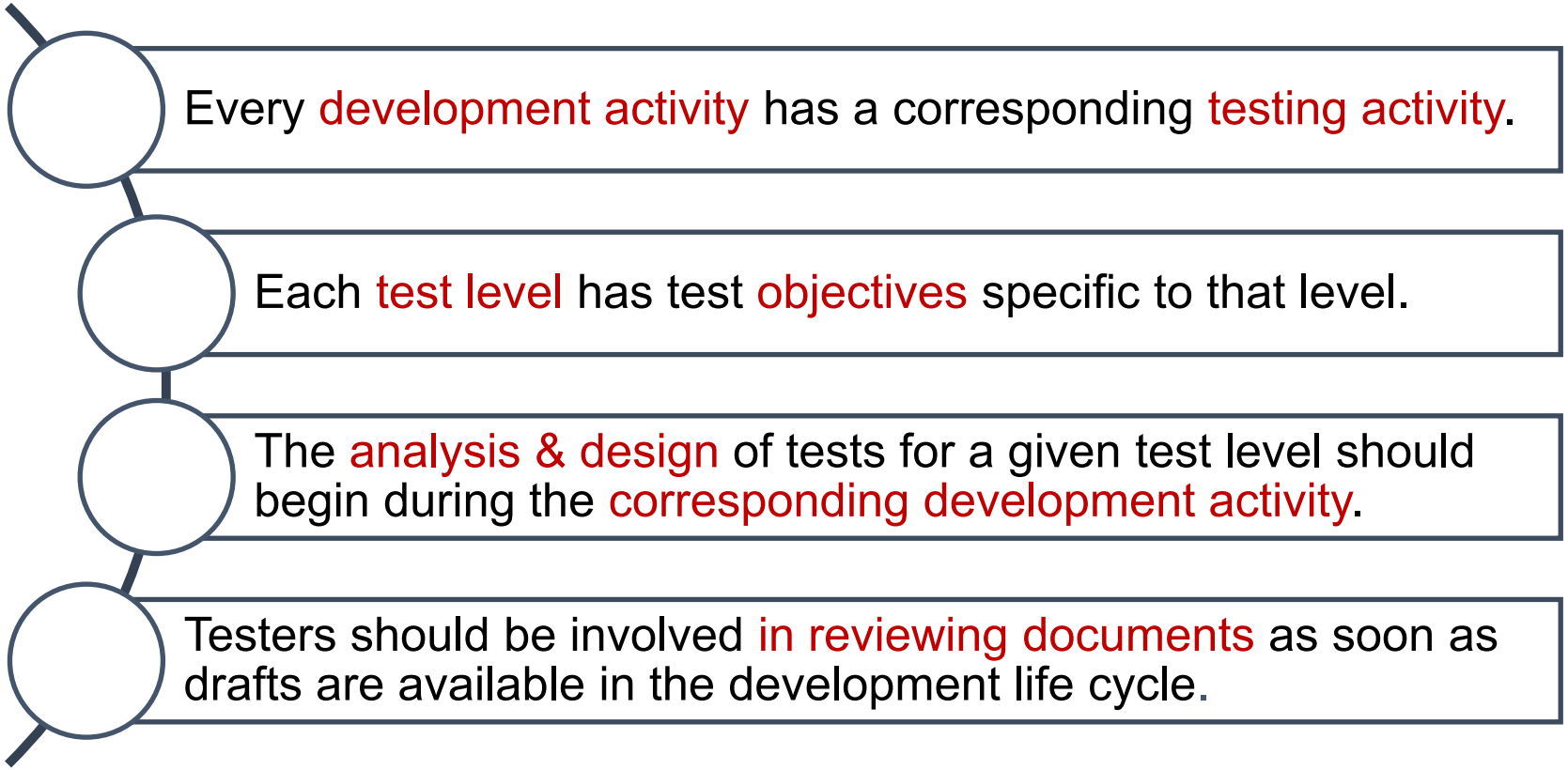
1. Software development models
- *1.1 Sequential model*
- *1.2 Iterative-incremental model*
- *1.3 Testing within a life-cycle model*

2. Test Levels
- *2.1 Component testing*
- *2.2 Integration testing*
- *2.3 System testing*
- *2.4 Acceptance testing*

3. Test Types
- *3.1 Testing of function*
- *3.2 Testing of non-functional software characteristics*
- *3.3 Testing of software structure*
- *3.4 Testing related to changes*

4. Maintenance testing

- In any life cycle model, there are several characteristics of good testing:



- Test levels can be combined or reorganized depending on the nature of the project or the system architecture

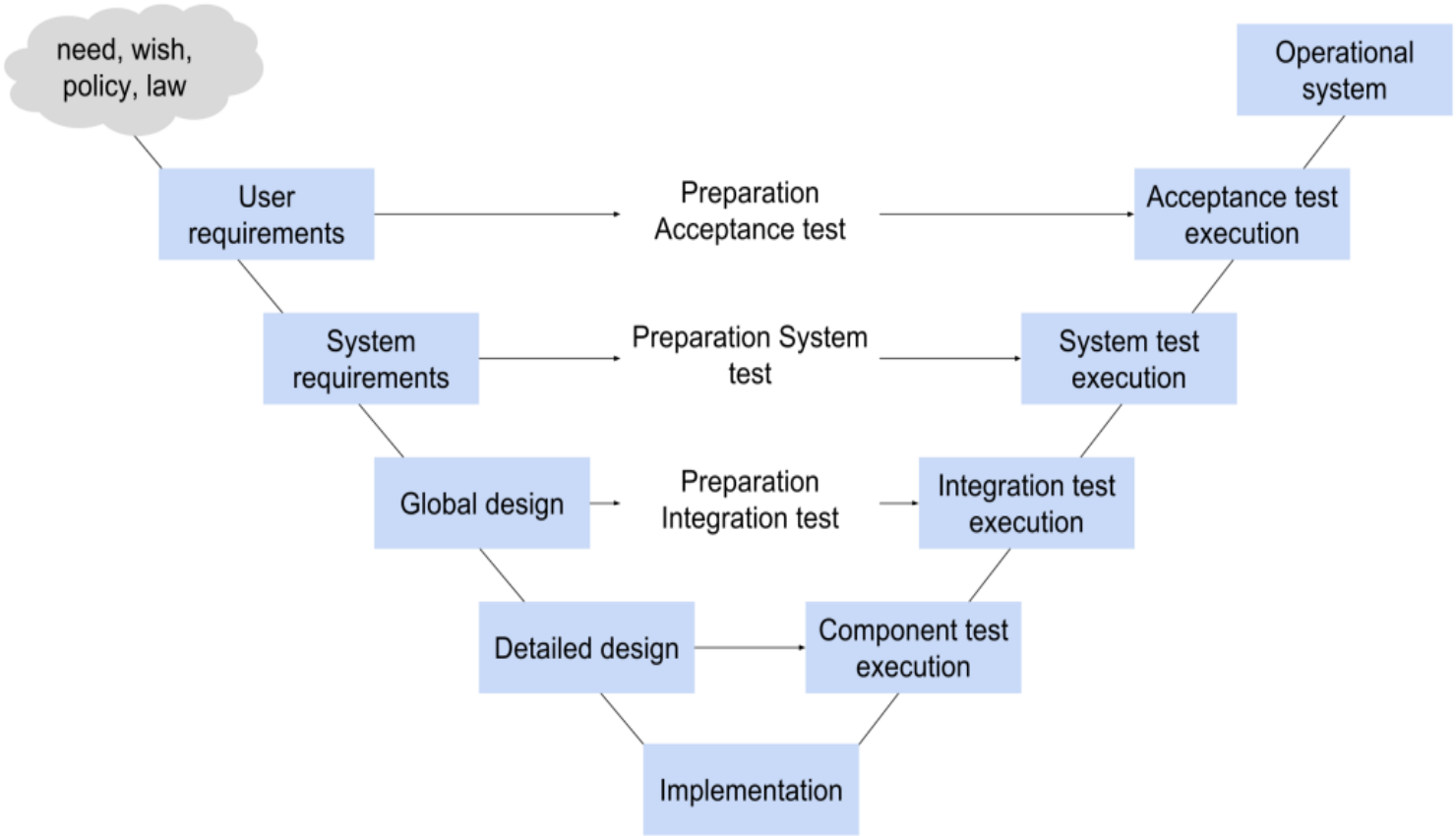
Testing within a life cycle model

1. *Software development models*
- *1.1 Sequential model*
- *1.2 Iterative-incremental model*
- *1.3 Testing within a life-cycle model*

2. *Test Levels*
- *2.1 Component testing*
- *2.2 Integration testing*
- *2.3 System testing*
- *2.4 Acceptance testing*

3. *Test Types*
- *3.1 Testing of function*
- *3.2 Testing of non-functional software characteristics*
- *3.3 Testing of software structure*
- *3.4 Testing related to changes*

4. *Maintenance testing*



1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

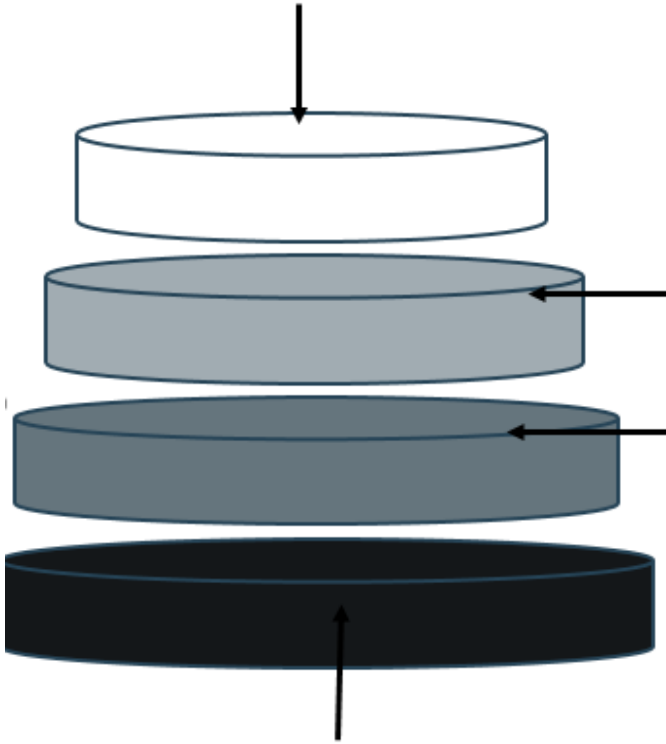
3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Test Levels

- **Acceptance**

  Is the responsibility of the customer – in general. The goal is to gain confidence in the system; especially in its non-functional characteristics



- **System**

  The behavior of the whole product(system) as defined by the scope of the project

- **Integration**

  Interface between components; interactions with other systems (OS, HW, …)

- **Unit**

  Any module, program, object separately testable

Component testing - objectives

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

- Verifying the functioning of software items (modules, methods, objects, classes etc.) that are separately testable

- Component testing includes testing of
  - functionality
  - specific non-functional characteristics, such as:
    - resource-behavior (e.g. memory leaks)
    - robustness testing

Component testing – test basis

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

All materials that are applicable for the component under test:

- Specification of the component
- Software design
- The data model
- As well as the code itself

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Component testing

- Writing code to test the project code
- Stubs, drivers and simulators may be used.

Stubs:

Code that replaces a called component in order to simulate its purpose (i.e. "hard-coded" data to replace data from a database).

Drivers:

Code that replaces another software component in order to call the component under test.

Component testing - example

1. *Software development models*
- *1.1 Sequential model*
- *1.2 Iterative-incremental model*
- *1.3 Testing within a life-cycle model*

2. *Test Levels*
- *2.1 Component testing*
- *2.2 Integration testing*
- *2.3 System testing*
- *2.4 Acceptance testing*

3. *Test Types*
- *3.1 Testing of function*
- *3.2 Testing of non-functional software characteristics*
- *3.3 Testing of software structure*
- *3.4 Testing related to changes*

4. *Maintenance testing*


```
public void getLocationListTest()
{
    //Arrange
    VisitorRepositoryStub stub = new VisitorRepositoryStub();
    VisitorBLL bll = new VisitorBLL(stub);
    List<LocationSummary> expectedResult = stub.getLocationList("nb-NO");
    //Act
    List<LocationSummary> result = bll.getLocationList("nb-NO");
    //Assert
    Assert.Equal(expectedResult.Count(), result.Count());
    Assert.Equal(expectedResult[0].id, result[0].id);
}
```

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Component testing – approaches and responsibilities

Component testing usually involves the programmer who wrote the code.

Defects are fixed as soon as they are found, without formal recording of incidents.

TDD test-driven development

- prepare and automate test cases before coding
- used in XP (extreme programming)

Integration testing - objectives

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

- Tests interfaces between components
- Test interactions with different parts of a system, i.e. interfaces between modules, such as
  - GUI
  - database
  - program logic
  - the operating system
  - file system
  - hardware

Integration testing – test basis

1. Software development models
- *1.1 Sequential model*
- *1.2 Iterative-incremental model*
- *1.3 Testing within a life-cycle model*

2. Test Levels
- *2.1 Component testing*
- *2.2 Integration testing*
- *2.3 System testing*
- *2.4 Acceptance testing*

3. Test Types
- *3.1 Testing of function*
- *3.2 Testing of non-functional software characteristics*
- *3.3 Testing of software structure*
- *3.4 Testing related to changes*

4. Maintenance testing

- Software and system design
- The system architecture
- data flow
- Workflows/use cases

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Integration testing – test objects

The item under test includes

- builds including some or all component of the system
- The database elements
- System infrastructure
- Interface between components or objects
- System configuration
- Configuration data

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

NB! Two "new" levels in 5 Edition

Component integration

Tests the interactions between software components is done after component testing (previously ~ integration testing)

System integration

Tests the interactions between different systems is done after system testing (new level)

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Integration testing – approaches and responsibilities

- Start integration with those components that are expected to cause most problems.
- To reduce the risk of late defect discovery, integration should normally be incremental rather than “big bang”.
- Both functional and structural approaches may be used.
- Ideally, testers should understand the architecture and influence integration planning.
- Can be done by the developers or by a separate team.

System testing – objectives

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Testing the behavior of the whole system as defined by the scope of the project.

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

System testing – test basis

- System requirements specification, both functional and non-functional
- Business processes
- Risk analysis
- Use cases
- Other high-level descriptions of the system behavior, interactions with OS/system resources
- Requirements may exist as text and/or models.
- Testers also need to deal with incomplete or undocumented requirements.

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

System testing – test objects

- The entire integrated system
- User manuals
- Operation manuals
- System configuration information
- Configuration data

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

System testing – approaches and responsibilities

Test environment should correspond to the production environment as much as possible.

- First, the most suited black-box technique
- Then, white-box technique to assess the thoroughness of testing

An independent test team may be responsible for the testing

The level of independence is based on the applicable risk level

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Acceptance testing - objectives

The questions to be answered:

- *Can the system be released?*
- *What are the outstanding risks?*
- *Has development met its obligations?*

The goal is to establish confidence in

- The system
- Non-functional characteristics of the system

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Acceptance testing - types

- **User acceptance testing –** validate the fitness for use of the system by users.

- **Operational testing, usually done by the system administrators:**
  - testing of backup/restore
  - disaster recovery
  - user management
  - maintenance tasks
  - periodic checks of security vulnerabilities

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Acceptance testing - types

- **Contract and regulation acceptance testing** performed against a contract's acceptance criteria (i.e. governmental, legal or safety regulations)

- **Alpha and beta testing**
  - Alpha testing is performed at the developing organization's site.
  - Beta testing (field testing), is performed by people at their own locations.

  Both are performed by potential customers, not the developers of the product.

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Acceptance testing - responsibilities

- Customers or users of a system
- Stakeholders may be involved

Test Types

1. *Software development models*
   - *1.1 Sequential model*
   - *1.2 Iterative-incremental model*
   - *1.3 Testing within a life-cycle model*

2. *Test Levels*
   - *2.1 Component testing*
   - *2.2 Integration testing*
   - *2.3 System testing*
   - *2.4 Acceptance testing*

3. *Test Types*
   - *3.1 Testing of function*
   - *3.2 Testing of non-functional software characteristics*
   - *3.3 Testing of software structure*
   - *3.4 Testing related to changes*

4. *Maintenance testing*

- Functional
- Non-Functional
- Black-Box
- White-box
- (used to be structural + related to change)

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Functional testing

Objectives

Test what a system should do and consider the external behavior of the software.

Test levels

May be performed at all test levels

Test basis

The expected behavior description can be found in work products such as:

- requirements specification
- business processes
- use cases
- functional specifications
- may be undocumented

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Non-functional testing

Objectives

Measuring characteristics of software that can be quantified on a varying scale: e.g. response times for performance testing

Test levels

May be performed at all test levels

You can find more about them in ‘Software Engineering – Software Product Quality’ (ISO 9126).

Software Product Quality

1. *Software development models*
- *1.1 Sequential model*
- *1.2 Iterative-incremental model*
- *1.3 Testing within a life-cycle model*

2. *Test Levels*
- *2.1 Component testing*
- *2.2 Integration testing*
- *2.3 System testing*
- *2.4 Acceptance testing*

3. *Test Types*
- *3.1 Testing of function*
- *3.2 Testing of non-functional software characteristics*
- *3.3 Testing of software structure*
- *3.4 Testing related to changes*

4. *Maintenance testing*



Black-box **testing**

1. *Software development models*
- *1.1 Sequential model*
- *1.2 Iterative-incremental model*
- *1.3 Testing within a life-cycle model*

2. *Test Levels*
- *2.1 Component testing*
- *2.2 Integration testing*
- *2.3 System testing*
- *2.4 Acceptance testing*

3. *Test Types*
- *3.1 Testing of function*
- *3.2 Testing of non-functional software characteristics*
- *3.3 Testing of software structure*
- *3.4 Testing related to changes*

4. *Maintenance testing*

Specification based testing.

Focuses on testing a software application without requiring knowledge of the internal code or implementation.

Can be applied at all ISTQB levels.

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Structural testing (White-box)

Objectives

Measuring the thoroughness of testing through assessment of the coverage of a set of structural elements or coverage items.

Test levels

May be performed at all test levels, but especially in component testing and component integration testing.

Structural testing (White-box)

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Objectives

Measuring the thoroughness of testing through assessment of the coverage of a set of structural elements or coverage items.

Test levels

May be performed at all test levels, but especially in component testing and component integration testing.

Structural testing

1. *Software development models*
- *1.1 Sequential model*
- *1.2 Iterative-incremental model*
- *1.3 Testing within a life-cycle model*

2. *Test Levels*
- *2.1 Component testing*
- *2.2 Integration testing*
- *2.3 System testing*
- *2.4 Acceptance testing*

3. *Test Types*
- *3.1 Testing of function*
- *3.2 Testing of non-functional software characteristics*
- *3.3 Testing of software structure*
- *3.4 Testing related to changes*

4. *Maintenance testing*

Test basis

Structural testing is based on the structure of the code as well as the architecture of the system (e.g. a calling hierarchy, a business model or a menu structure)

Approach

Structural techniques are best used *after specification-based* techniques, in order to help measure the thoroughness of testing.

Tools

Coverage measurement tools assess the percentage of executable elements (e.g. statements or decision outcomes) that have been exercised.

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Testing related to changes, confirmation and regression testing

Confirmation testing

After a defect is detected and fixed, the software should be retested to confirm that the original defect has been successfully removed.

Regression testing

The repeated testing of an already tested program, after modification, to discover any defects introduced or uncovered as a result of the change(s).

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Testing related to changes

Objective

To verify that modifications in the software or environment have not caused unintended side effects and that the system still meets its requirements.

Test levels

May be performed at all test levels, applies to functional, non-functional and structural testing.

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Testing related to changes

Objective

To verify that modifications in the software or environment have not caused unintended side effects and that the system still meets its requirements.

Test levels

May be performed at all test levels, applies to functional, non-functional and structural testing.

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Testing related to changes

Approach

The extent of regression testing is based on the risk of finding defects in software that was working previously.

Regression test suites are run many times and generally evolve slowly, so regression testing is a strong candidate for automation.

If the regression test suite is very large it may be more appropriate to select a subset for execution.

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Testing related to changes

Approach

The extent of regression testing is based on the risk of finding defects in software that was working previously.

Regression test suites are run many times and generally evolve slowly, so regression testing is a strong candidate for automation.

If the regression test suite is very large it may be more appropriate to select a subset for execution.

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Testing related to changes

Approach

The extent of regression testing is based on the risk of finding defects in software that was working previously.

Regression test suites are run many times and generally evolve slowly, so regression testing is a strong candidate for automation.

If the regression test suite is very large it may be more appropriate to select a subset for execution.

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Maintenance testing

Objectives

Maintenance testing is done on an existing operational system, and is triggered by modifications, migration, or retirement of the software or system.

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Maintenance testing - types

Modifications

- Planned enhancement changes (release-based)
- Corrective and emergency changes (patches)
- Changes of environment (operating system or database upgrades)

Migration (from one platform to another)

- operational tests of the new environment
- tests on the changed software

Retirement of a system

The testing of data migration or archiving if long data-retention periods are required

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Maintenance testing - types

Modifications

- Planned enhancement changes (e.g. release-based)
- Corrective and emergency changes (patches)
- Changes of environment (operating system or database upgrades)

Migration (e.g. from one platform to another)

- operational tests of the new environment
- tests on the changed software

Retirement of a system

The testing of data migration or archiving if long data-retention periods are required.

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Maintenance testing

Scope

The scope of maintenance testing is related to:

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

Test levels

Depending on the changes, maintenance testing may be done at any or all test levels and for any or all test types.

1. Software development models
- 1.1 Sequential model
- 1.2 Iterative-incremental model
- 1.3 Testing within a life-cycle model

2. Test Levels
- 2.1 Component testing
- 2.2 Integration testing
- 2.3 System testing
- 2.4 Acceptance testing

3. Test Types
- 3.1 Testing of function
- 3.2 Testing of non-functional software characteristics
- 3.3 Testing of software structure
- 3.4 Testing related to changes

4. Maintenance testing

Maintenance testing

Approach

Determining how the existing system may be affected by changes is called impact analysis and is used to help decide how much regression testing to do.

Note

Maintenance testing can be difficult if specifications are out of date or missing.