ISTQB – Foundation Level

CHAPTER 2: TESTING THROUGHOUT THE SOFTWARE LIFE CYCLY

Prepared by: Vu Nguyen

April 2010

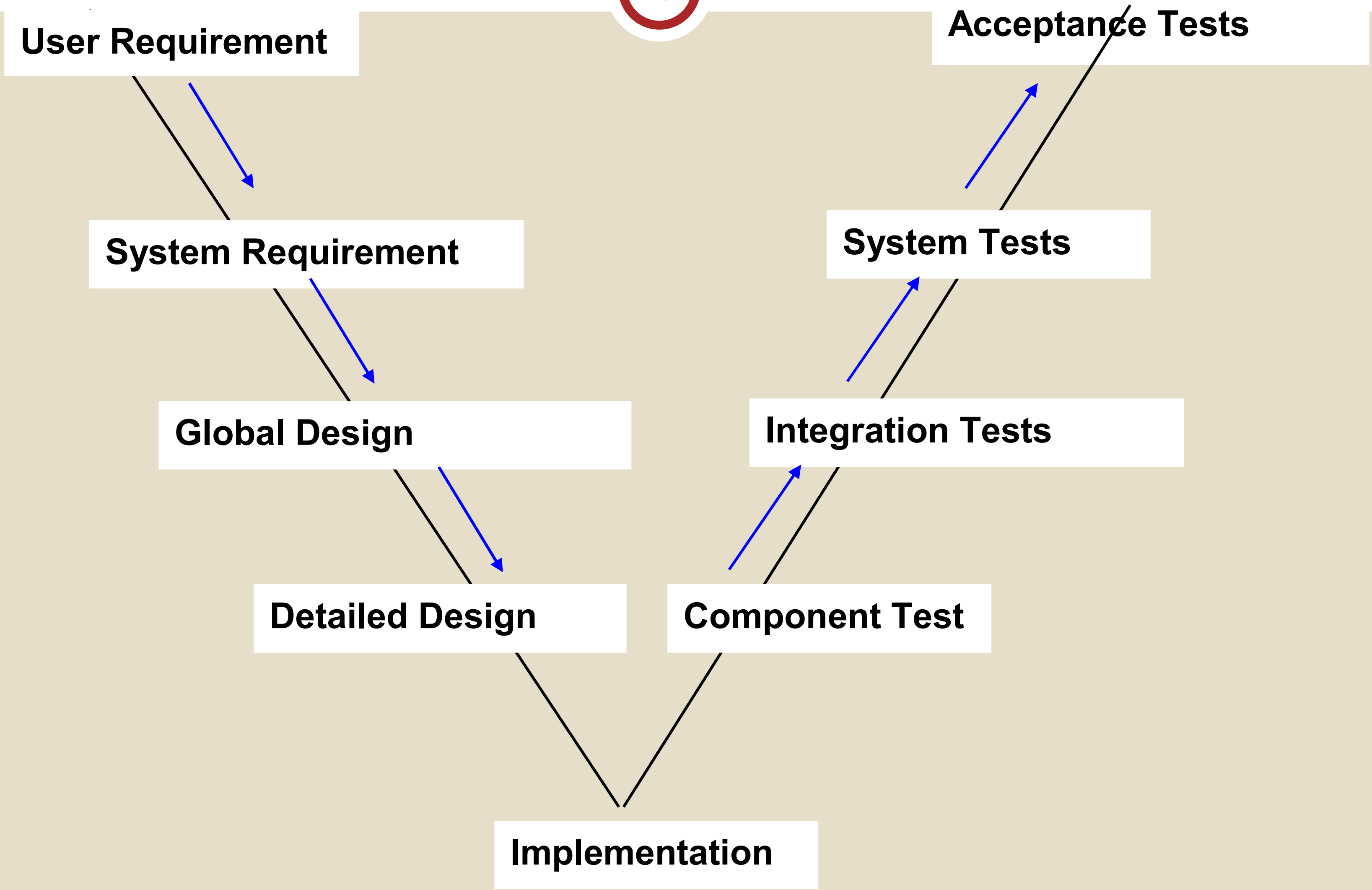
AGENDA

- 2.1 Software development models (K2)
- 2.2 Test levels (K2)
- 2.3 Test types (K2)
- 2.4 Maintenance testing (K2)

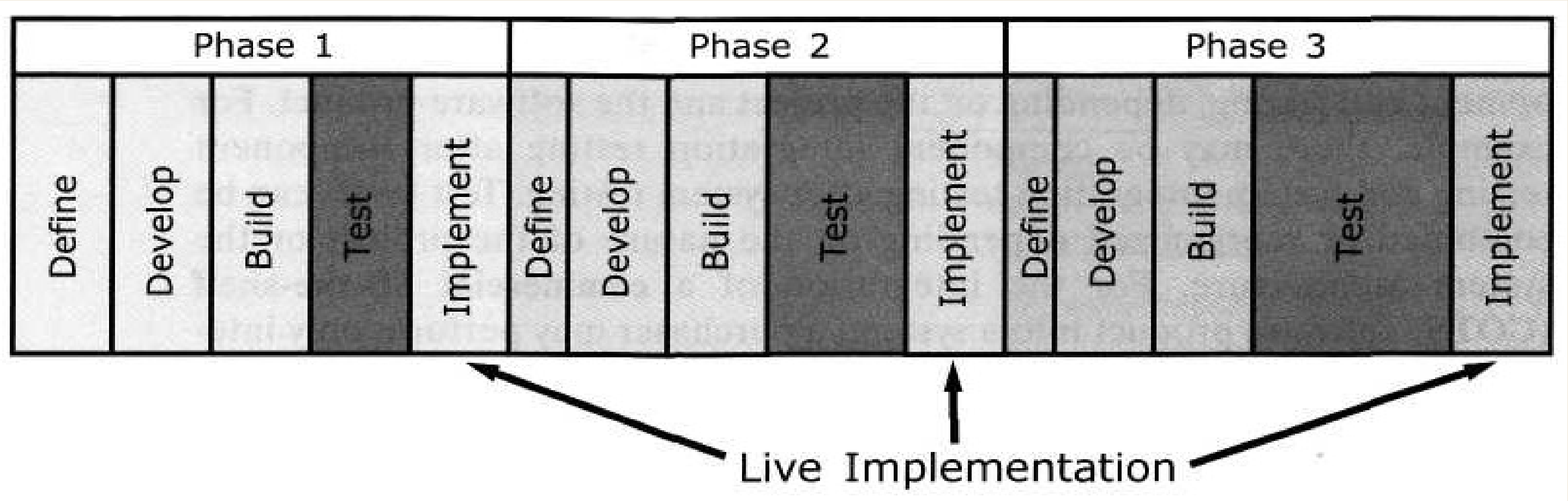
Objectives

- 2.1 Software development models (K2)
  - LO-2.1.1 Understand the relationship between development, test activities and work products in the development life cycle, and give examples based on project and product characteristics and context (K2).
  - LO-2.1.2 Recognize the fact that software development models must be adapted to the context of project and product characteristics. (K1)
  - LO-2.1.3 Recall reasons for different levels of testing, and characteristics of good testing in any life cycle model. (K1)

2.1.1 V-model (K2)



2.1.2 Iterative-incremental (K2)



2.1.2 Iterative-incremental (K2)

- We cycle through a number of smaller self-contained life cycle phases for the same project.
- The delivery is divided into increments or builds with each increment adding new functionality.
- Advantages:
  - Early market presence with critical functionality
  - Can reduce initial investment → Cost more in the long run.
  - Early market presence will mean validation testing is carried out and giving early feedback on the business value

2.1.2 Iterative-incremental (K2)

- Examples of iterative or incremental development models
  - Prototyping,
  - Rapid Application Development (RAD),
  - Rational Unified Process (RUP) and
  - Agile development.

2.1.2 Iterative-incremental (K2)

- Prototyping Model
  - The Prototyping Model is a systems development method (SDM) in which a prototype is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed.
  - This model works best in scenarios where not all of the project requirements are known in detail ahead of time.
  - It is an iterative, trial-and-error process that takes place between the developers and the users.

2.1.2 Iterative-incremental (K2)

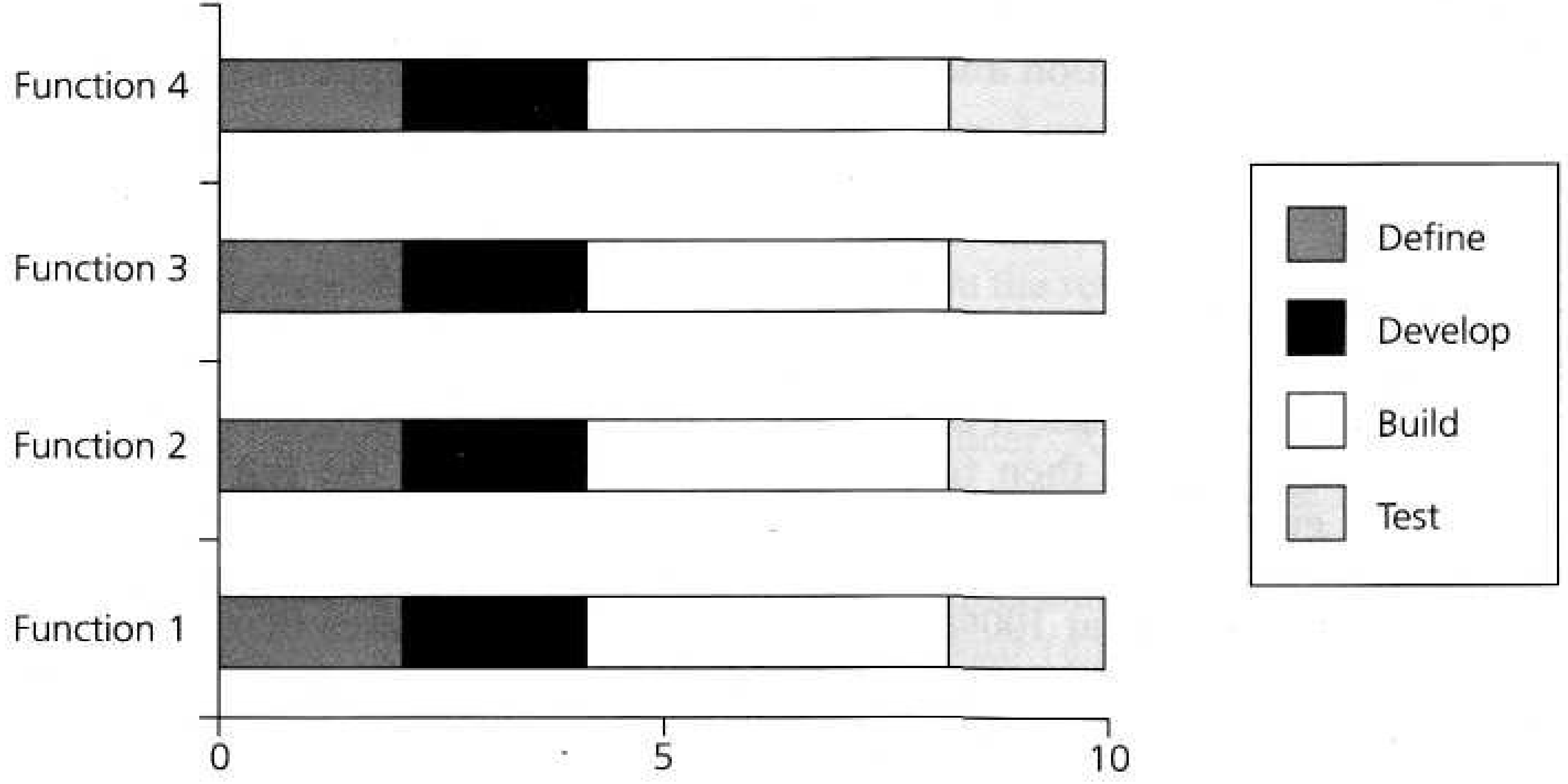
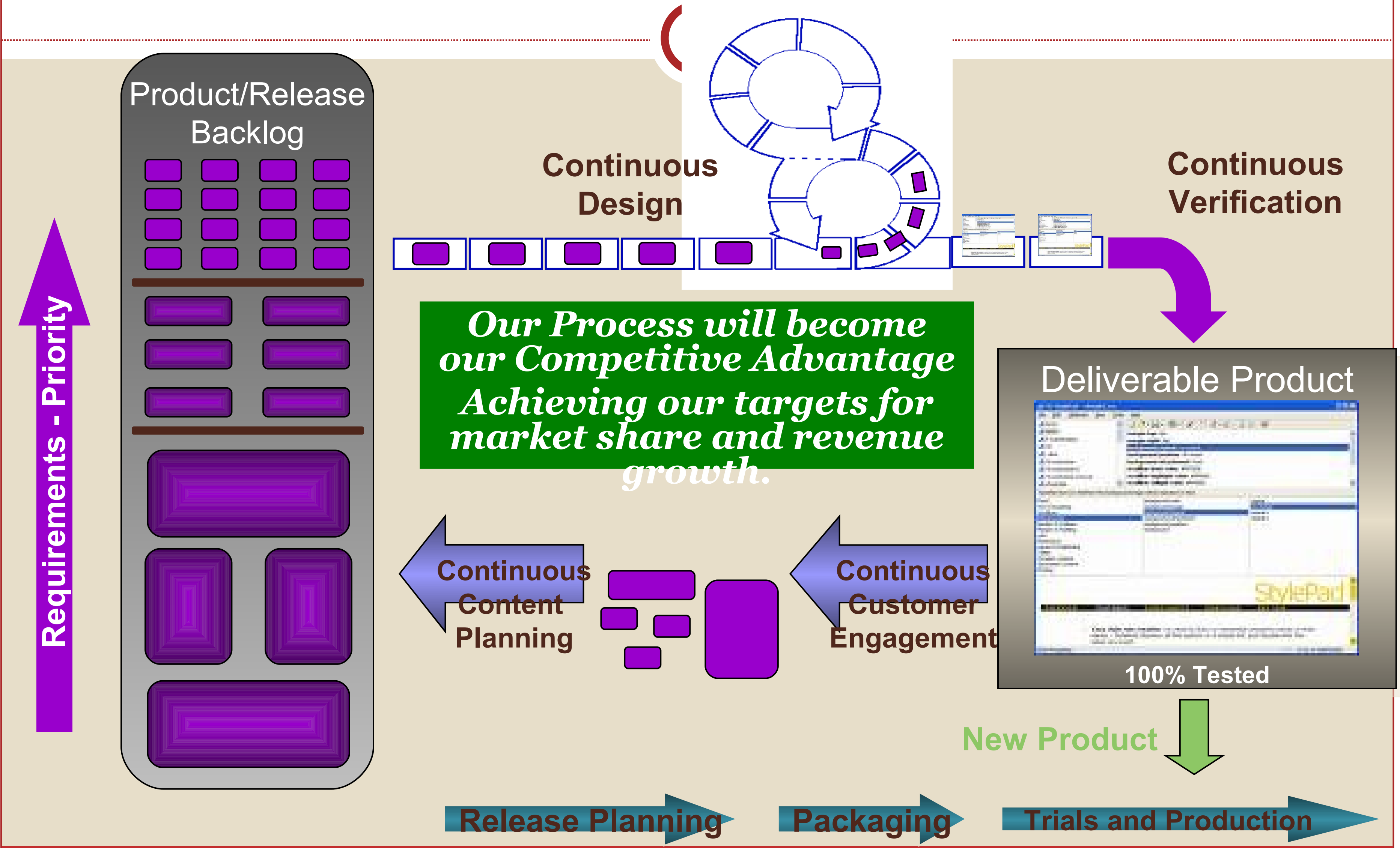


FIGURE 2.4 RAD model

2.1.2 Iterative-incremental (K2)

- Agile development
  - Agile chooses to do things in small increments with minimal planning, rather than long-term planning
  - Iterations are short time frames which typically last from one to four weeks.
  - Each iteration is worked on by a team including planning, requirement analysis, design, coding, unit testing, and acceptance testing when a working product is demonstrated to stakeholders.
  - This helps to minimize the overall risk, and allows the project to adapt to changes quickly.

The Agile Development Ecosystem



Product/Release Backlog

Requirements - Priority

Continuous Design

Continuous Verification

Our Process will become our Competitive Advantage Achieving our targets for market share and revenue growth.

Deliverable Product

100% Tested

Continuous Content Planning

Continuous Customer Engagement

New Product

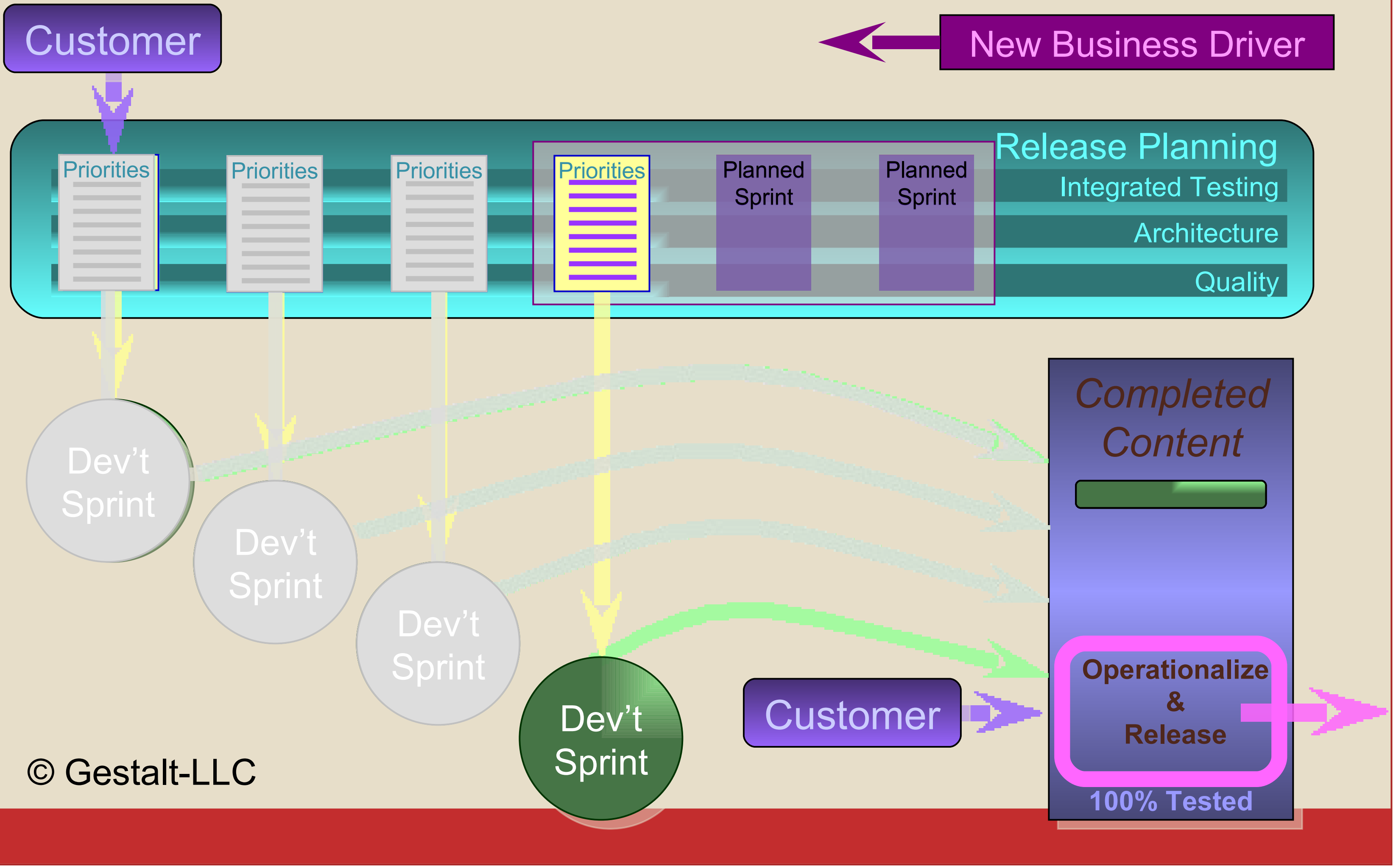
Release Planning

Packaging

Trials and Production

Adapted from: “Introduction to Agile Software Development with Scrum/XP”, Agile Infusion

Scrum Product Release View



© Gestalt-LLC

2.1.3 Testing within a life cycle model (K2)

- In summary, whichever life cycle model is being used, there are several characteristics of good testing:
  - For every development activity there is a corresponding testing activity;
  - Each test level has test objectives specific to that level;
  - 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 cycle.

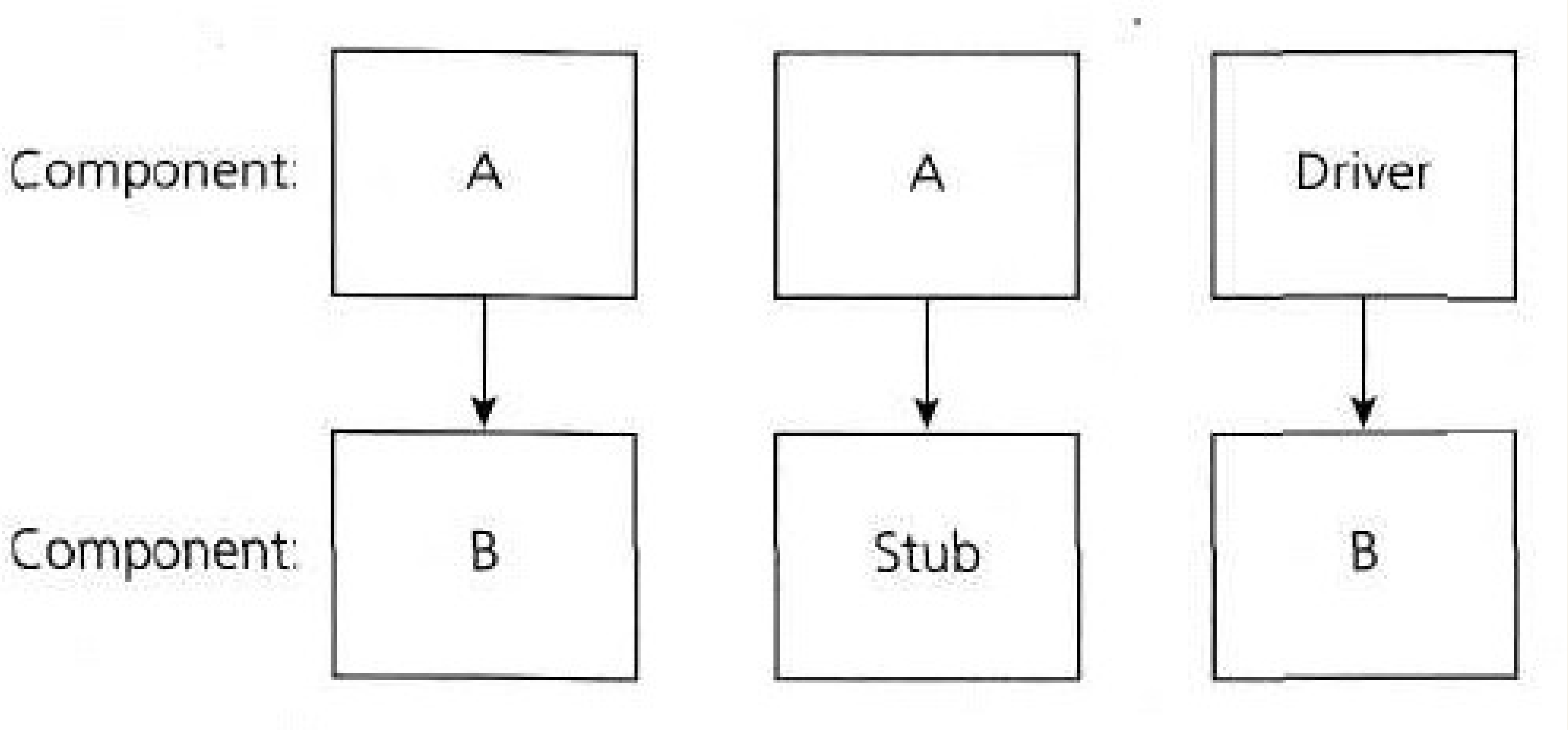
Test Levels

- 2.2 Objectives
  - LO-2.2.1 Compare the different levels of testing: major objectives, typical objects of testing, typical targets of testing (e.g. functional or structural) and related work products, people who test, types of defects and failures to be identified. (K2)

2.2.1 Component testing (K2)

- Component testing, also known as unit, module and program testing, searches for defects in, and verifies the functioning of software (e.g. modules, programs, objects, classes, etc.) that are separately testable.
- Developer is responsible for the unit test
- Devices: Stub, driver

2.2.1 Component testing (K2)



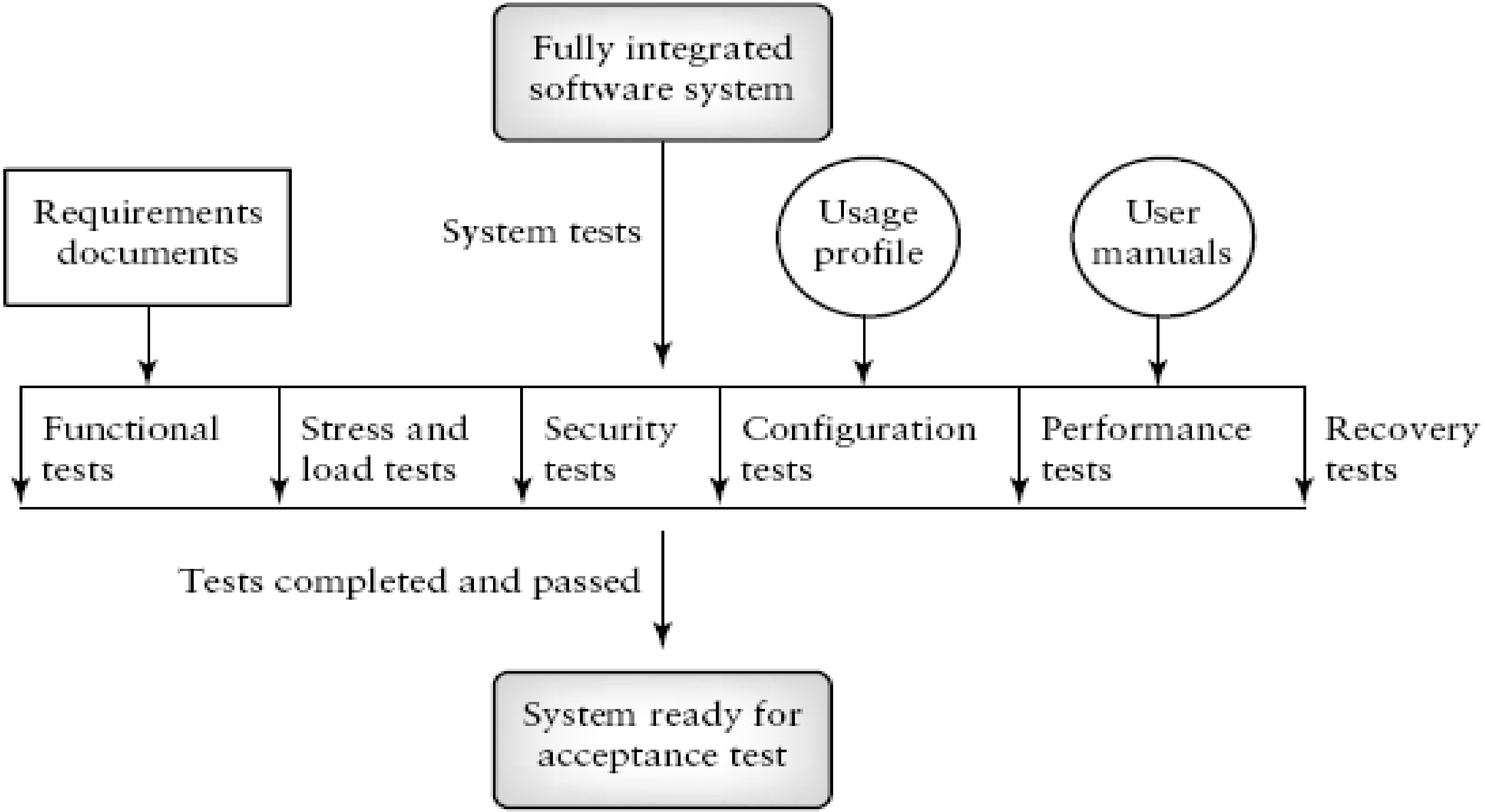
2.2.2 Integration testing (K2)

- Integration testing tests interfaces between components, interactions with different parts of a system.
- Approach:
  - Big-Bang
  - Top-down method
  - Bottom-up method
  - Funtional incremental

2.2.3 System testing (K2)

- System testing of software is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.
- System testing falls within the scope of black box testing.
- System testing should investigate both functional and non-functional requirements
- The test environment should be similar to the target environment as much as possible in order to minimize the risk.

2.2.3 System testing (K2)



2.2.4 Acceptance testing (K2)

- Formal testing with respect to user needs, requirements and business.
- Acceptance testing is most often the responsibility of the user or customer
- The goal of acceptance testing is to establish confidence in the system, part of the system or specific non-functional characteristics, e.g. usability, of the system.
- Finding defects should not be the main focus in acceptance testing.

2.2.4 Acceptance testing (K2)

- Some acceptance testing types:
  - User acceptance testing
  - Operational (acceptance) testing
    - Testing of backup/restore;
    - Disaster recovery;
    - User management;
    - Maintenance tasks;
    - Periodic checks of security vulnerabilities.
  - Contract and regulation acceptance testing
  - Alpha and beta (or field) testing

Test Types

- 2.3 Objectives (K2)
  - LO-2.3.1 Compare four software test types (functional, non-functional, structural and change related) by example. (K2)
  - LO-2.3.2 Recognize that functional and structural tests occur at any test level. (K1)
  - LO-2.3.3 Identify and describe non-functional test types based on non-functional requirements.(K2)
  - LO-2.3.4 Identify and describe test types based on the analysis of a software system's structure or architecture. (K2)
  - LO-2.3.5 Describe the purpose of confirmation testing and regression testing. (K2)

2.3 Test types (K2)

- Functional Testing
  - The function of a system (component)
    - = What it does?
  - Functional testing: Testing based-on an analysis of the specification of the functionality of a component or system.
  - Functionality is the capability of the software product to provide functions that meet stated and implied needs.
  - Functionality testing: The process of testing to determine the functionality of a software product.

2.3 Test types (K2)

- Functional Testing
  - Functionality testing can be done on:
    - Requirements-based testing
    - Business-process-based testing
  - Additional functional testing:
    - Suitability
    - Interoperability
    - Security
    - Accuracy and compliance

2.3 Test types (K2)

- Non Functional Testing
  - Non-functional attributes: Quality characteristics
  - Non-functional Testing is test how well or how fast something is done.

2.3 Test types (K2)

- Non Functional Testing
  - Performance testing
  - Load testing
  - Stress testing
  - Usability testing
  - Maintainability testing
  - Portability testing

2.3 Test types (K2)

- Structural Testing
  - Structural testing = white-box or glass-box
  - Structural testing: Testing based on an analysis of the internal structure of the component or system
  - Mostly applied at component & integration testing
  - Structure-based techniques(1) (white-box techniques) are used for structural testing
    - (1) A procedure to derive or select TCs based on an analysis of the internal structure of a component or system.

2.3 Test types (K2)

- Testing related to changes:
  - Confirmation testing
    - Retesting to verify the success of corrective action
  - Regression Testing
    - Checking the system has not regressed
    - To verify that modifications in the software or the environment have not caused unintended adverse side effects and that the system still meets its requirement.

Maintenance testing

- 2.4 Objectives (K2)
  - LO-2.4.1 Compare maintenance testing (testing an existing system) to testing a new application with respect to test types, triggers for testing and amount of testing. (K2)
  - LO-2.4.2 Identify reasons for maintenance testing (modification, migration and retirement). (K1)
  - LO-2.4.3. Describe the role of regression testing and impact analysis in maintenance. (K2)

2.4 Maintenance testing (K2)

- Maintenance Testing
  - Maintenance: Modification of a software product after delivery.
  - Maintenance Testing: Maintenance testing is done on an existing operational system.
  - Maintainability: How easy it is to modify the system.
  - Maintainability Testing: The process of testing to determine the maintainability of a software product.

2.4 Maintenance testing (K2)

- Impact Analysis
  - Maintenance testing consist of two parts:
    - Testing the changes
    - Regression Testing
  - Impart Analysis: A decision is made on what parts need careful regression testing.

2.4 Maintenance testing (K2)

- Triggers for maintenance testing
  - Triggers for maintenance:
    - Modification
    - Migration
    - Retirement (Data migration or archiving)

2.4 Maintenance testing (K2)

- Triggers for maintenance testing
  - Planned modification:
    - Perfect modification
    - Adaptive modification
    - Corrective planned modification
  - Ad-hoc corrective modifications
    - Defects requiring an immediate solution
    - Risk analysis should be performed.

Summary

- Software Development Model
  - V model
  - Iterative life cycles
- Test Levels
  - Component Testing
  - Integration Testing
  - System Testing
  - Acceptance Testing

Summary

- Test Types
  - Functional Testing
  - Non-functional Testing
  - Structural Testing
  - Confirmation & Regression Testing
- Maintenance Testing
  - Impact analysis
  - Triggers for maintenance Testing

References

- Rex Black, Foundations of Software Testing
- ISTQB Foundation Syllabus.pdf

Q & A

Glossary

- **Load testing:** A type of performance testing conducted to evaluate the behavior of a component or system with increasing load, e.g. numbers of parallel users and/or numbers of transactions, to determine what load can be handled by the component or system.
- **Performance testing:** The process of testing to determine the performance of a software product. See also efficiency testing.

Glossary

- **Stress testing:** A type of performance testing conducted to evaluate a system or component at or beyond the limits of its anticipated or specified work loads, or with reduced availability of resources such as access to memory or servers. [After IEEE 610] See also performance testing, load testing.
- **Efficiency testing:** The process of testing to determine the efficiency of a software product.

Glossary

- **Suitability:** The capability of the software product to provide an appropriate set of functions for specified tasks and user objectives. [ISO 9126] See also functionality.
- **Functional testing:** Testing based on an analysis of the specification of the functionality of a component or system. See also black box testing.

Glossary

- **Stub:** 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]
- **Driver:** A software component or test tool that replaces a component that takes care of the control and/or the calling of a component or system. [After TMap]

Glossary

- **Interoperability:** The capability of the software product to interact with one or more specified components or systems. [After ISO 9126] See also functionality.
- **Usability testing:** Testing to determine the extent to which the software product is understood, easy to learn, easy to operate and attractive to the users under specified conditions. [After ISO 9126]

Glossary

- **Maintainability:** The ease with which a software product can be modified to correct defects, modified to meet new requirements, modified to make future maintenance easier, or adapted to a changed environment. [ISO 9126]
- **Portability:** The ease with which the software product can be transferred from one hardware or software environment to another. [ISO 9126]

Glossary

- **Non-functional testing:** Testing the attributes of a component or system that do not relate to functionality, e.g. reliability, efficiency, usability, maintainability and portability.
- **Confirmation testing:** See re-testing.
- **Re-testing:** Testing that runs test cases that failed the last time they were run, in order to verify the success of corrective actions.

Glossary

- **Regression testing:** Testing of a previously tested program following modification to ensure that defects have not been introduced or uncovered in unchanged areas of the software, as a result of the changes made. It is performed when the software or its environment is changed.
- **Reliability:** The ability of the software product to perform its required functions under stated conditions for a specified period of time, or for a specified number of operations. [ISO 9126]