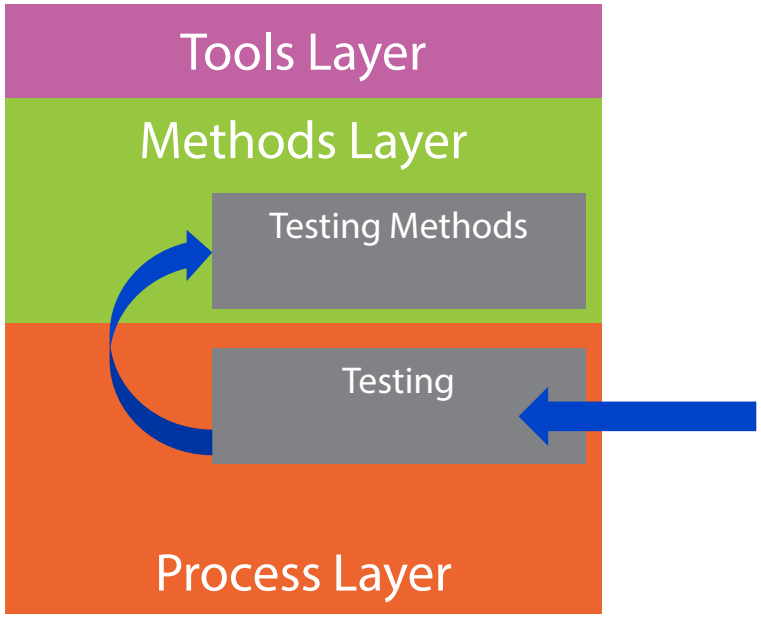
Testing

Mohamad Halabi
@mohamadhalabi

pluralsight
hardcore dev and IT training

Introduction



Testing vs. Quality Control vs. Quality Assurance

- **Quality Assurance (QA)**
  - Not related to testing
  - It's about processes, practices, and standards
  - SWEBOK: "*is a set of activities that define and assess…that the software processes are appropriate and produce software products of suitable quality for their intended purposes.*"
    - Process definition
    - Process measurement
    - Continuous improvement

Testing vs. QC

- **SWEBOK: "*SQC activities examine specific project artifacts (documents and executables) to determine whether they comply with…requirements, constraints, designs, contracts, and plans*"**
- **QC examines both documents and executables**
  - Scope of QC is not only software
- **QC examines outputs against requirements, design, constraints, contracts, and plans**
  - Test cases can verify software against requirements, design, and constraints
  - Contracts and plans require other techniques, such as reviews
  - Therefore QC includes activities such as reviews and inspections

Testing vs. QC

Testing

Concerned with software

One of QC activities

QC

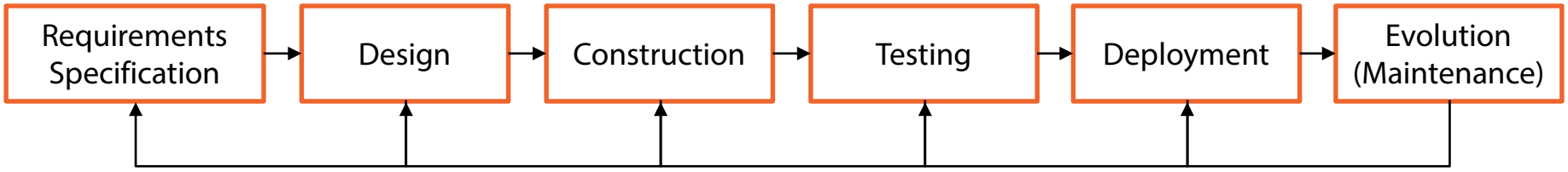
Concerned with all project artifacts including software

Includes testing in addition to other techniques, such as reviews and inspections

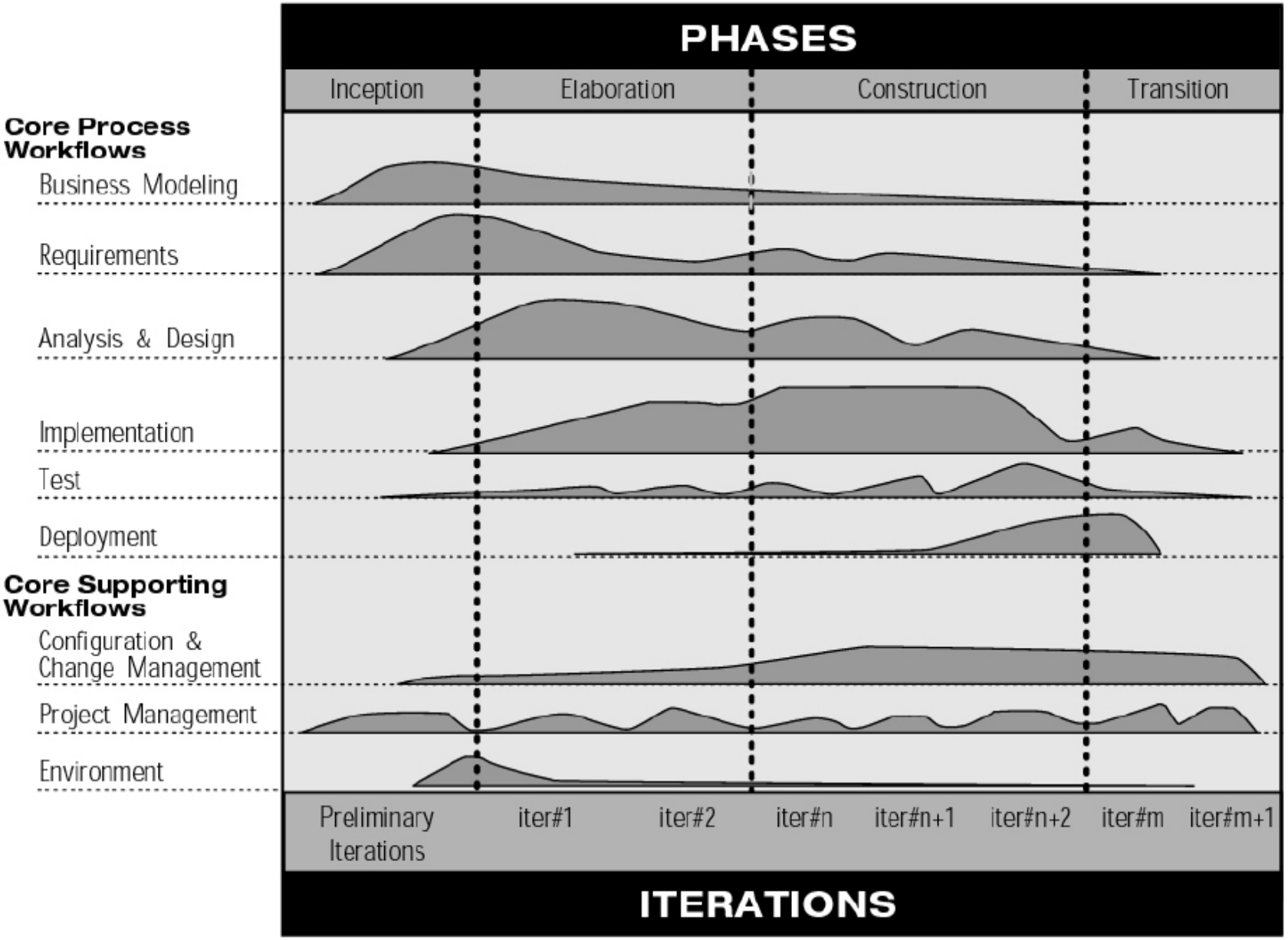
From Construction to Testing

- Construction includes subset of testing activities (unit and integration)
- Relationship between Construction and Testing phases depends on the selected process model

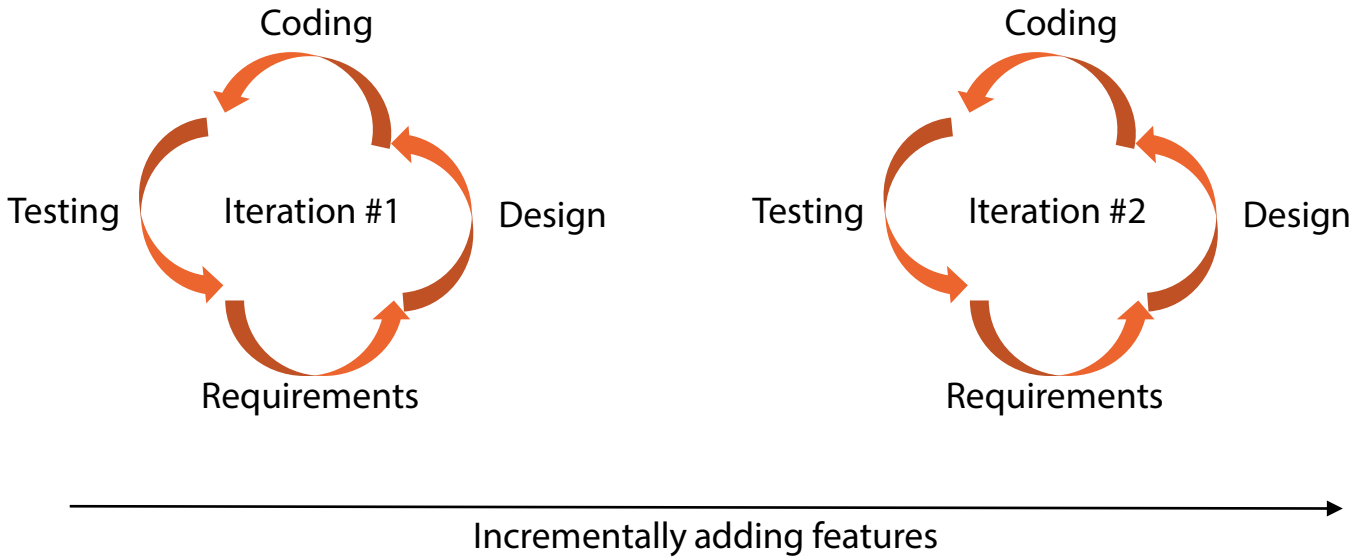
Waterfall


Requirements Specification
Design
Construction
Testing
Deployment
Evolution (Maintenance)

RUP


PHASES
Inception
Elaboration
Construction
Transition
Core Process Workflows
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Core Supporting Workflows
Configuration & Change Management
Project Management
Environment
Preliminary Iterations
iter#1
iter#2
iter#n
iter#n+1
iter#n+2
iter#m
iter#m+1
ITERATIONS

Agile


Coding
Testing
Iteration #1
Design
Requirements
Coding
Testing
Iteration #2
Design
Requirements
Incrementally adding features

Common Principles

- **Regardless of the process model, testing principles are the same**
- **Process models impact activities**
  - Flow
  - Emphasis

Common Definitions

- **It's important to agree on common terminology across teams**

Term	Definition
Defect	• Work product deficiency • Work product does not meet its specifications • Talk here is about "work products" in general
Fault	• A defect in source code • Incorrect step, process, or data definition • If 'work product' == 'software' then Fault == Defect
Bug	• Fault is the formal definition of bug • Fault and bug can be used interchangeably
Failure	• System or component does not perform functions within limits • Caused by faults

- **Differentiate between:**
  - Cause of malfunction; i.e. fault, defect, or bug
  - Undesired effect in system's service; i.e. failure
- **Testing reveals failures; the causing faults must be repaired**

Test Plan

- **Plans the entire testing effort:**
  - Scope
  - Deliverables
  - Roles and responsibilities
  - Tools
  - Testing method
  - Activities
  - Schedule
  - Success criteria
  - Risk assessment
- **Frames and controls testing**

Test Scenario

- **Summarized description for testing a requirement**
- **Example test scenario for a Reservation use case:**
  - "Validate that only logged-in users can confirm a reservation and purchase tickets"
- **Test scenarios are about what to test, rather than how to test it**

Test Case

- **Single test scenario might result in multiple test cases**
- **A test case is a step-by-step sequence for achieving a test scenario outcome**
  - Sequence of testing steps
  - Preconditions
  - Input data
  - Expected output
  - Actual output
  - Post conditions

Test Case

Related Test Scenario ID	TS001			
Preconditions	• User has already navigated to the site • User has already searched for a flight			
		Expected Result	**Actual Result**	**Status**
Step1	Click on flight to reserve	Redirect to login page	Redirect to login page	Succeed
Step2	Enter valid <username>	Field must be editable	Value accepted	Succeed
Step3	Enter valid <password>	Password must not be shown in clear text	Password appears in clear format	Fail
Step4	Click Login	Login accepted and redirected to flights page	Login accepted, but redirected to home page	Fail

Test Script

- **Opinion 1: A run of a test case generated by specific data**

Related Test Scenario ID	TS001			
Preconditions	• User has already navigated to the site • User has already searched for a flight			
		Expected Result	**Actual Result**	**Status**
Step1	Click on flight to reserve	Redirect to login page	Redirect to login page	Succeed
Step2	Enter valid <username> (myusername)	Field must be editable	Value accepted	Succeed
Step3	Enter valid <password> (mypassword)	Password must not be shown in clear text	Password appears in clear format	Fail
Step4	Click Login	Login accepted and redirected to flights page	Login accepted, but redirected to home page	Fail

Test Script

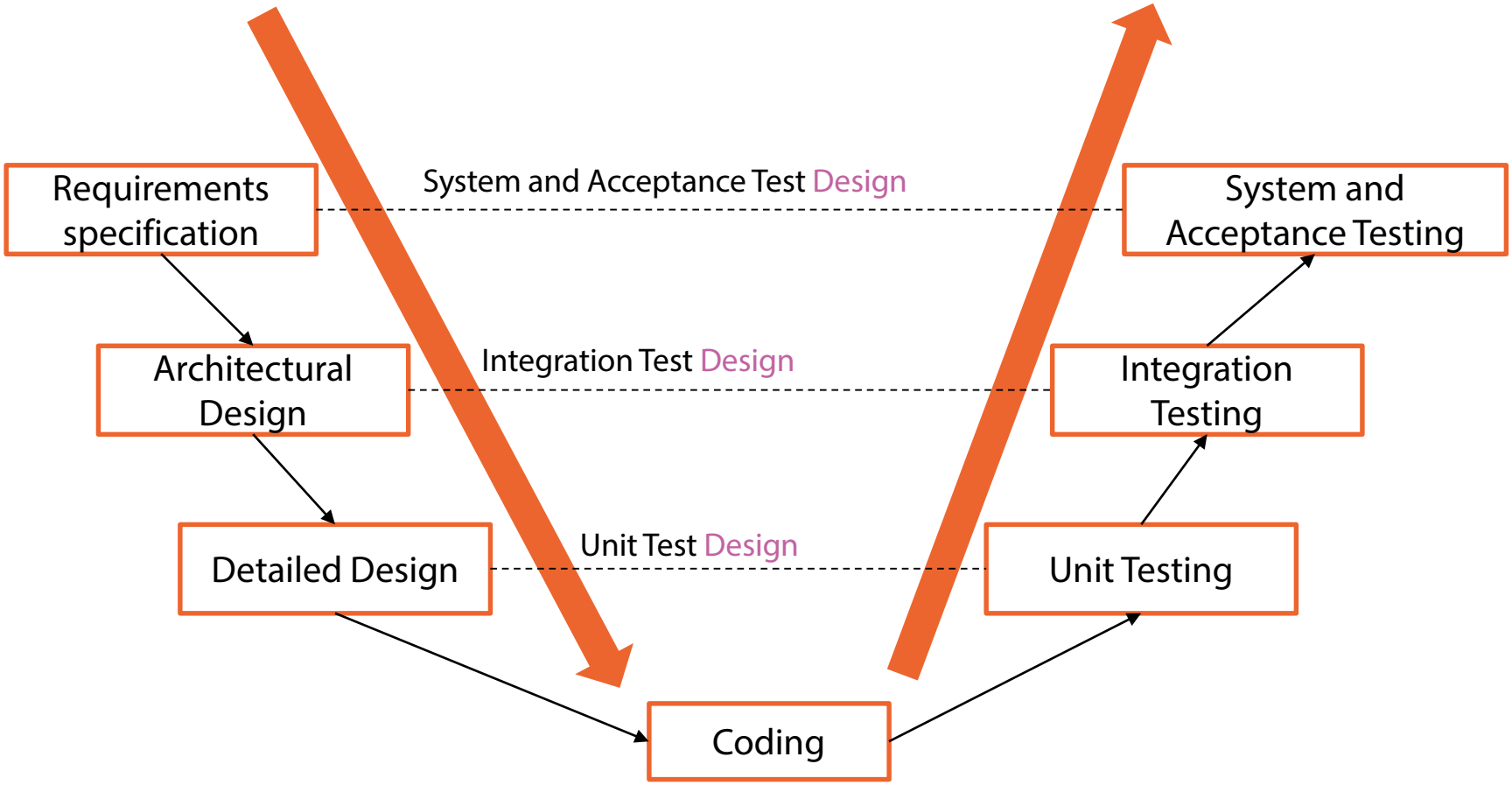
- **Opinion 2: Test scripts are automation scripts**
- **Interpreted by automation tools to automate the run of test cases**
  - Ex: VBScript

The V-Shaped Model

- **A software process model, just like Agile, Waterfall, etc…**
- **A variation of Waterfall**
  - Emphasis on testing
  - Testing is not an afterthought
  - Testing is planned through requirements to coding

The V-Shaped Model

- **V: Verification and Validation**



The V-Shaped Model

- **Verification and validation are required at each phase**

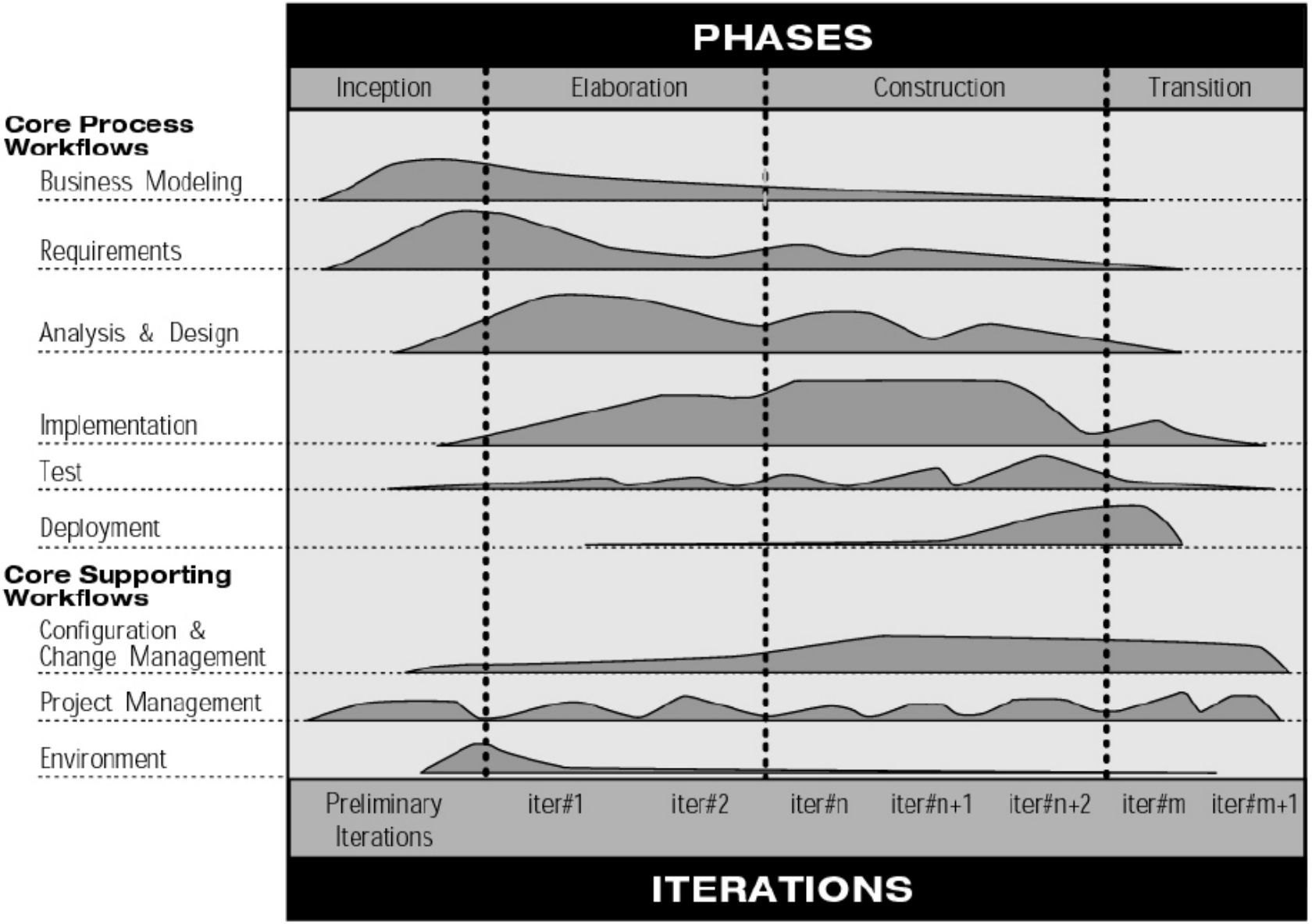
Verification	Validation
Ensures product is built correctly	Ensures right product is built
Output of an activity meets the specifications imposed by previous activities	Product fulfills its intended purpose (i.e. business value)
Product conforms to requirements and design	Product meets expectations
Ex: Design sufficient to implement requirements	Ex: Product satisfies user requirements (document)
Ex: Code correctly implements design	

- **Verification and validation can be done by static or dynamic techniques**
  - Static: Reviews and inspections
  - Dynamic: Testing methods

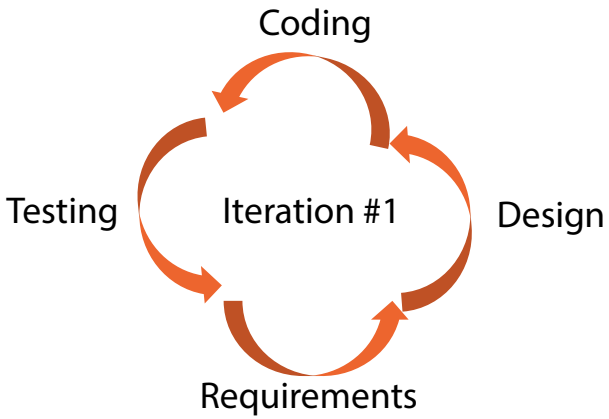
Usability of the V-Shaped Model

- **Test planning starts early in the lifecycle**
  - This increases quality of work products up until the final product
- **Q: Can we benefit from its advantages in iterative models?**
- **A: Yes, by adopting the practice behind it**
  - Do not adopt its sequential execution of activities
  - Make verification and validation (V&V) a planned approach
  - Any process model can adopt progressive planning of V&V within its activities (instead of a “big-bang” approach)

Usability of the V-Shaped Model



Usability of the V-Shaped Model



Test Techniques

- Different test techniques are used to detect failures
- Techniques discussed here are mentioned in SWEBOK

Specification-Based Testing

- **Also called Functional, Conformance, Black-Box, or Closed-Box testing**
- **Examines software against specifications without examining code details**

Technique	Description

Equivalence Partitioning	• Divides input data into set of "equivalence classes"
	• Example criteria: accepted and rejected data ranges
	• A set of test cases are designed for each class
	• A defect in one test case will likely be found in other cases of the same set
	• Testing coverage is maximized with minimum number of test cases
	• Example: Accept loans between $1 and $100
	  • Inputs 1→50 and 51→100 result in different approval processes
	  • Two equivalence classes are created (1→50) and (51→100)
	  • One set of test cases are created per class

Specification-Based Testing

Technique	Description

Boundary Value Analysis	• A form of Equivalence Partitioning
	• Data ranges selected are on and near boundaries of input domain
	• Many faults tend to concentrate near the extreme input values
	• Example:
	  • Test system against 1 and 100
	  • Also against 2 and 99

Robustness Testing	• Extension of Boundary Value Analysis • Test cases test data outside the input domain • Tests the system against unexpected inputs • Example: Test system's reaction for loan input of $-1

Code-Based Testing

- **Also called Structural, White-box, or Open-box testing**
- **Examines the internal code**

Technique	Description

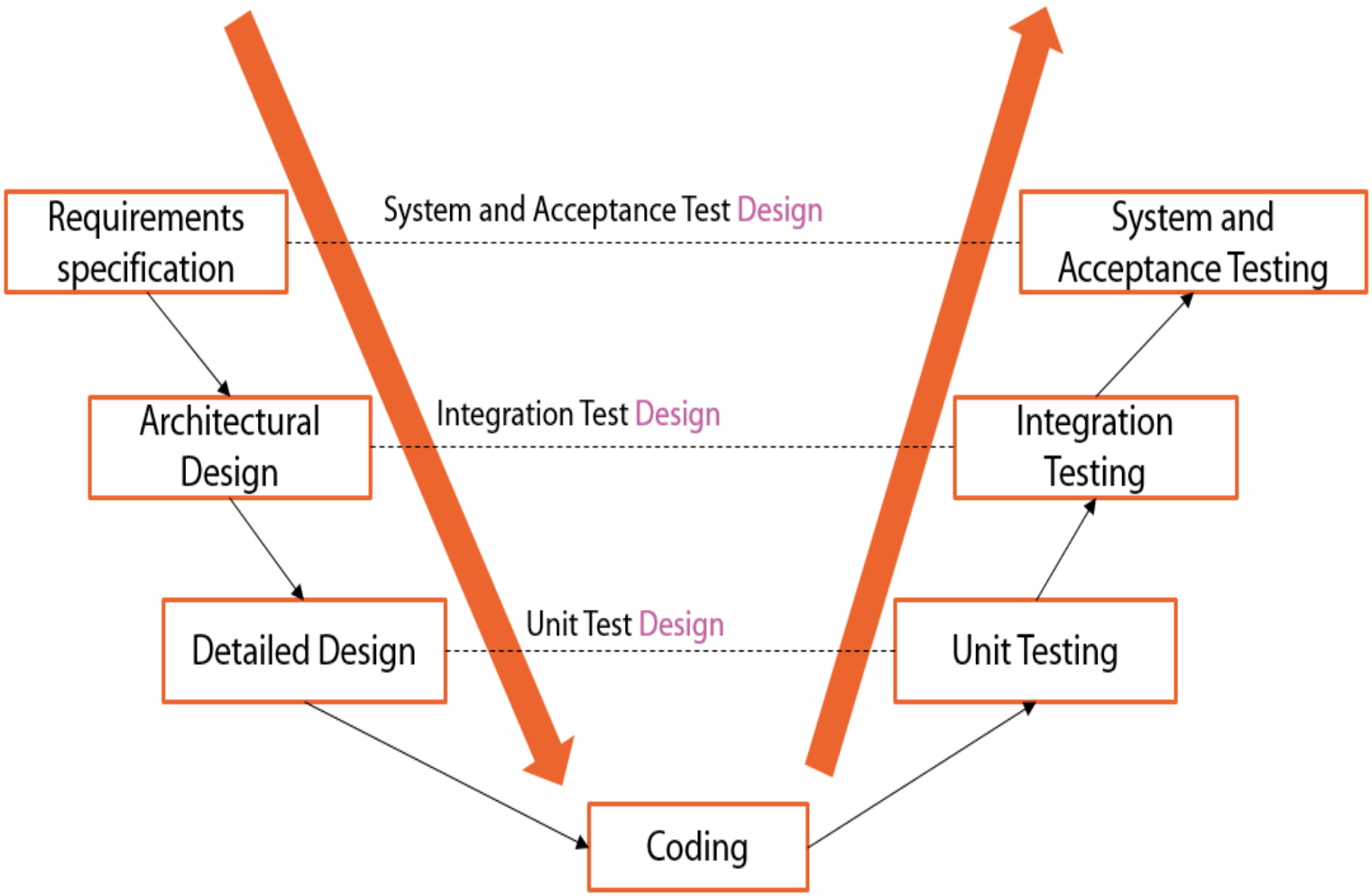
Control-Flow	• Covers all statements, statement blocks, or combination of the two
	• Derive a Flow Graph which represents the control flow
	• Flow Graph illustrates sequence, conditions, and loop statements
	  • Test all paths (not always practical due to loop iterations)
	  • Test selected paths and statements

Data-Flow	• Annotate the Flow Graph with information about variables definition, usage, and disposed • Look for risky patterns and design test cases accordingly • Ex: Using a variable after it has been disposed

Other Techniques

- **Based on a software engineer's intuition and experience**
  - Discover test cases not easily identified by formal techniques
- **Fault-based**
  - Reveals likely or predefined fault categories
  - Error guessing can be used to design test cases based on faults history
- **Application-specific**
  - Ex: GUI, protocol-conformance, and safety-critical formal testing

Test Levels


Requirements specification
System and Acceptance Test Design
System and Acceptance Testing
Architectural Design
Integration Test Design
Integration Testing
Detailed Design
Unit Test Design
Unit Testing
Coding

Unit Testing

- **Also called Module or Component testing**
- **Considered a coding practice**
- **Unit tests are written as soon as source code blocks must be tested**
- **Test Driven Development practice advocates writing unit tests before code**
- **Unit tests exercise non-executable code**
  - Drives (simulate calling unit) and stubs (simulate called unit) are needed

Integration Testing

- **Also called String testing**
- **Component interactions are tested against specifications**
- **Limited scope is done during construction**
- **Must be performed incrementally**
  - A “big-bang” approach – except for the simplest of programs – won’t work

Integration Testing

- **Strategies for integrating components:**

Strategy	Description
Top-down	• Starts from the top of the hierarchy (i.e. control module or main program) • Major components at the top of the hierarchy are tested first • Stubs are required to simulate components down the hierarchy
Bottom-up	• Starts at the bottom of the hierarchy • No need for stubs (lower-level components are available for testing) • The program as a unit only exists after the last component up in the hierarchy is tested
Combined (Sandwich Testing)	• Top-down for upper levels of program structure • Bottom-up for subordinate levels

System Testing

- **The complete and integrated system is tested**
- **System's compliance with specifications is tested**
- **Non-functional requirements are tested**
  - Must be performed on real infrastructure (hardware, network, etc…)

Acceptance Testing

- **Also called Pilot Testing**
- **Formal testing reveals if customers/users accept system/component**
- **In evolutionary development, apply testing on each increment**
  - Early feedback drives future increments

Test Objectives

- **Testing objective must be measurable in order to be controlled**

Objective	Description
Performance Testing	Verifies software meets performance requirements under expected load • Ex: Latency and throughput
Stress Testing	• Tests the software beyond expected load • Attempts to break the system to identify its limits under current resources
Recovery Testing	Verifies disaster recovery capabilities • Automatic recovery: re-initialization, data recovery, etc… • Manual recovery: Mean-Time-To-Repair (MTTR) by human intervention
Configuration Testing	Verifies all supported hardware and software configurations
Usability Testing	• Evaluates how easily users can operate, supply inputs, and interpret outputs for a system or component • Evaluates system's ability to recover from user errors

Test Objectives

Objective	Description
Installation Testing	• Verifies installation in the target environment • Equivalent to System testing in case of new environment
Alpha Testing	System is tested in-house by representatives of potential users (for trail)
Beta Testing	System is tested externally by representatives of potential users (for trail)
Regression Testing	• Selectively re-test the system or component to verify that modifications have not caused unintended effects • Tradeoffs must be done against cost of continuous re-testing

Summary

- **Testing is a Quality Control (QC) activity**
- **Test Plan plans the testing effort (scope, activities, resource, etc…)**
- **Test Case contains a detailed step of how to test**
  - Driven from a Test Scenario
- **Test Case design and execution must be a continuous effort**
  - The V-Shaped Model advocates not making testing an afterthought
- **Practice of the V-Shaped Model can be adopted in evolutionary models**

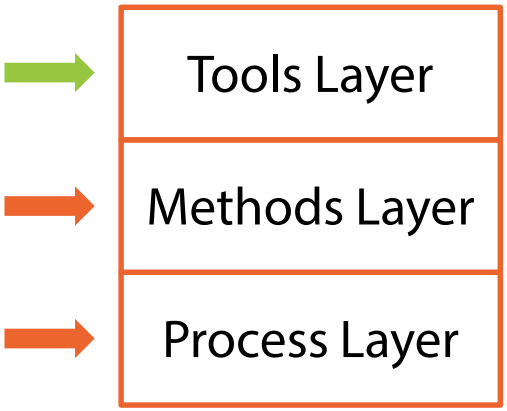
Summary

- **Test techniques**
  - Black-box
  - White-box
- **Test levels**
  - Unit testing
  - Integration testing
  - System testing
  - Acceptance testing
- **Test objectives**
  - Ex: Performance, usability, restore capability, configuration support, etc…

Where to Go from Here?

- **SWEBOK Knowledge Areas:**

1. Software Requirements
2. Software Design
3. Software Construction
4. Software Testing
5. Software Maintenance
6. Software Configuration Management
7. Software Engineering Management
8. Software Engineering Process
9. Software Engineering Models and Methods
10. Software Quality
11. Software Engineering Professional Practice
12. Software Engineering Economics
13. Computing Foundations
14. Mathematical Foundations
15. Engineering Foundations



- **SWEBOK Version 3**