

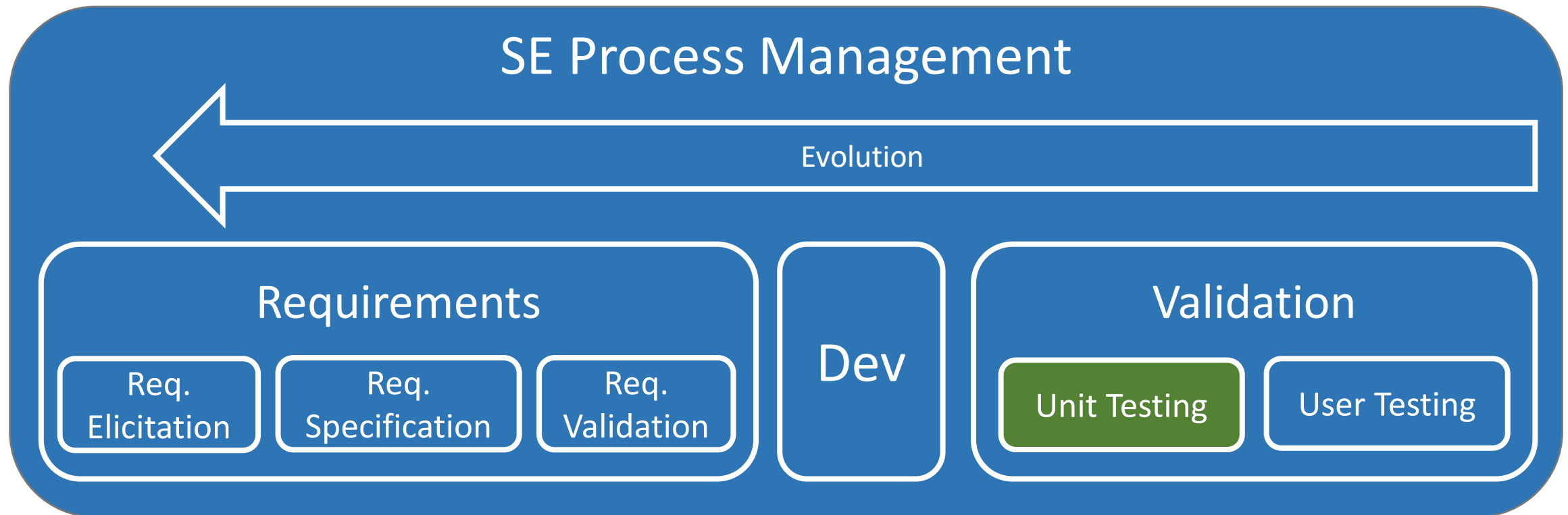
# Software Engineering COMP1035

## Lecture 10

*Software Validation & Testing*



# Keeping Track of SE Module



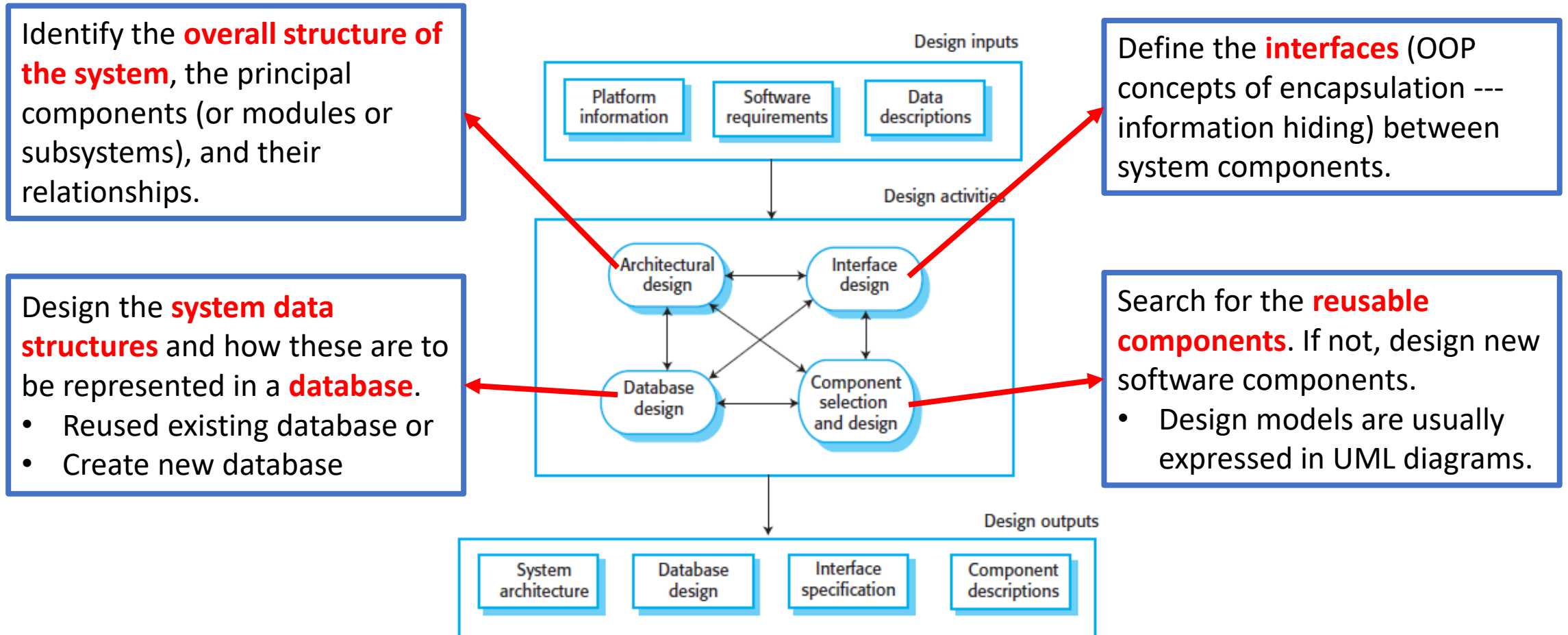


# Object-Oriented (OO) Design

---

What designs to be validated and verified in a software?

# General Model of Design Process



# How To Conduct Objected-Oriented Design

---

- Understand and **define the context** and the **external interactions** with the system.
- Design the **system architecture**.
- Identify the **principal objects** in the system
- Develop design models.
- Specify **interfaces**.





# Why Need Testing Plan?

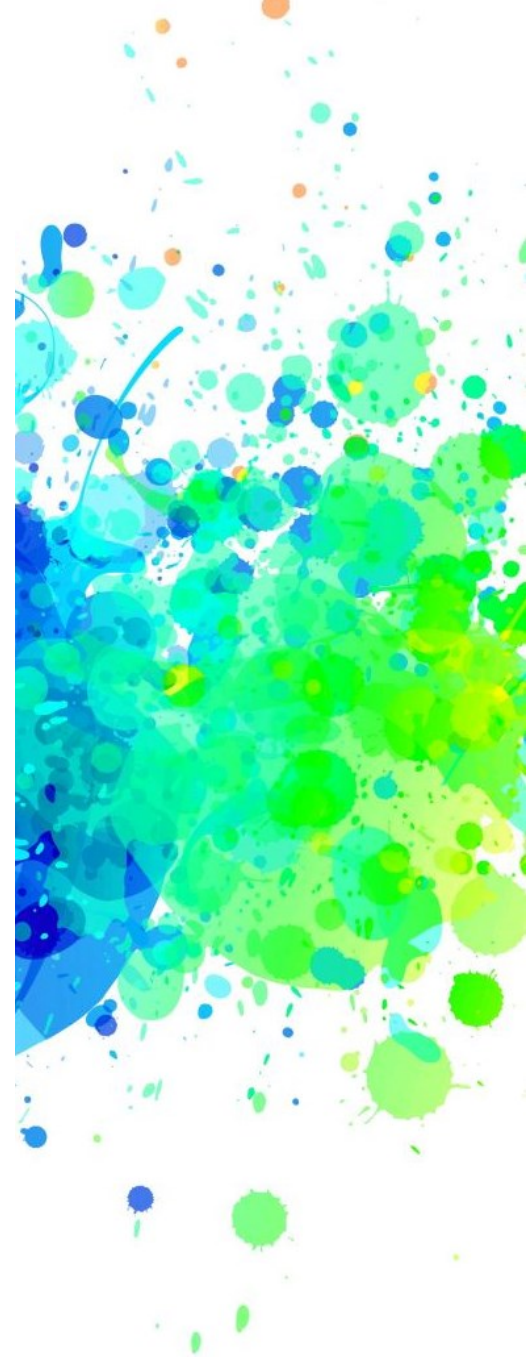
---

- We want to **have enough details** to give to the development team.
  - “Here, you build this” --- How do I know the development is correct?!
  - Not – here, please design this software.
- Creating a **final low-level design** is often led by a solution architect (SA)
  - Perhaps the most important tool in the toolbox is a visual documentation language, such as UML diagram.
  - In addition to UML, the SA (Solution Architect) may need to be good at database design.”
  - Perhaps the most critical skills for the SA are the **ability to create consensus and understanding around the architecture**.
- It is often the final version of all the design documents, read by the subsequent teams.



# Software Validation

---



# Software Validation & Verification

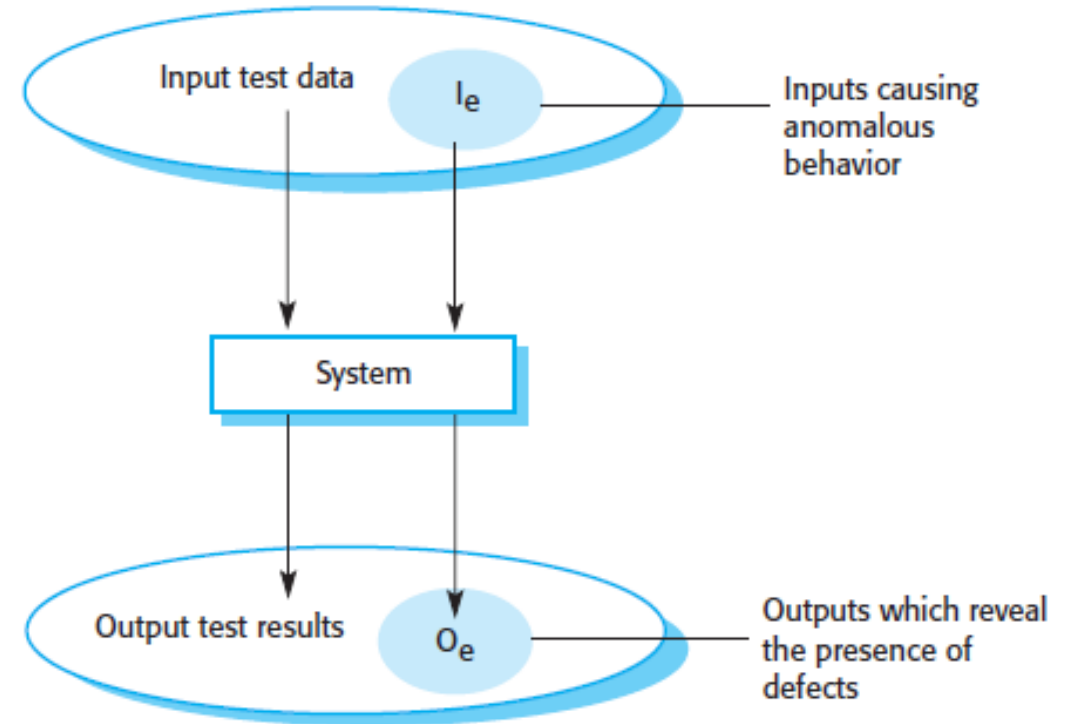
- Intended to verify that a system **fulfills its specifications** and **meets the expectations** of the customer.
- Program testing – using simulated test data is a principal validation technique.
- May also involve checking processes – **inspections and reviews** (happen from user requirements definition to system development).
- Three stages of testing process.



# Software V & V

---

- Verification:
  - Checking that the software **meets its stated functional and non-functional requirements**.
- Validation:
  - More general purposes.
  - Ensure software **meets the customer's expectations**.
  - Happens – when statements of requirements do not reflect the actual needs of the customers or users.



- **Validation:** *Are we building the right product?*
- **Verification:** *Are we building the product right?*

# Software V & V

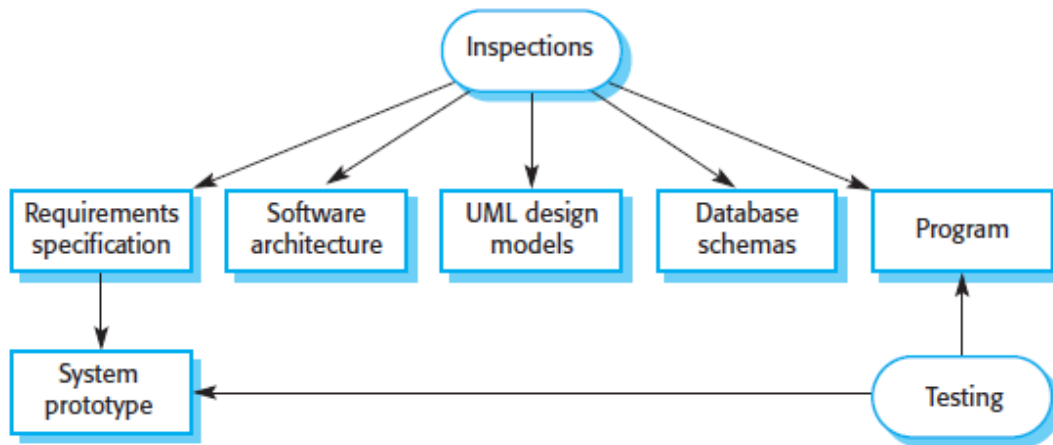
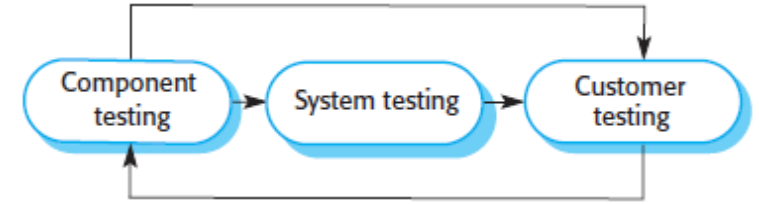


Fig: Inspections and Testing

- Analyse and check:
  - System requirements,
  - Design models,
  - Program source code,
  - Proposed system tests.
- Inspections **CANNOT** replace testing:
  - **Not good for discovering defects** – unusual interactions between different parts of a program, timing problems, or problems with system performance.

# Stages of Testing Process



- Component Testing:
  - Components are **tested by developers**.
  - Each component (functions or object classes) is tested independently.
- System Testing:
  - Finding **unusual interactions** between components and component interfaces.
  - To show system meets its functional and non-functional requirements.
- Customer Testing:
  - Final stage of testing before the system is accepted for use.
  - Tested by the customer rather than simulated test data.
  - May **reveal requirements problems** – do not meet users' needs or system performance is unacceptable.

# Test Plans

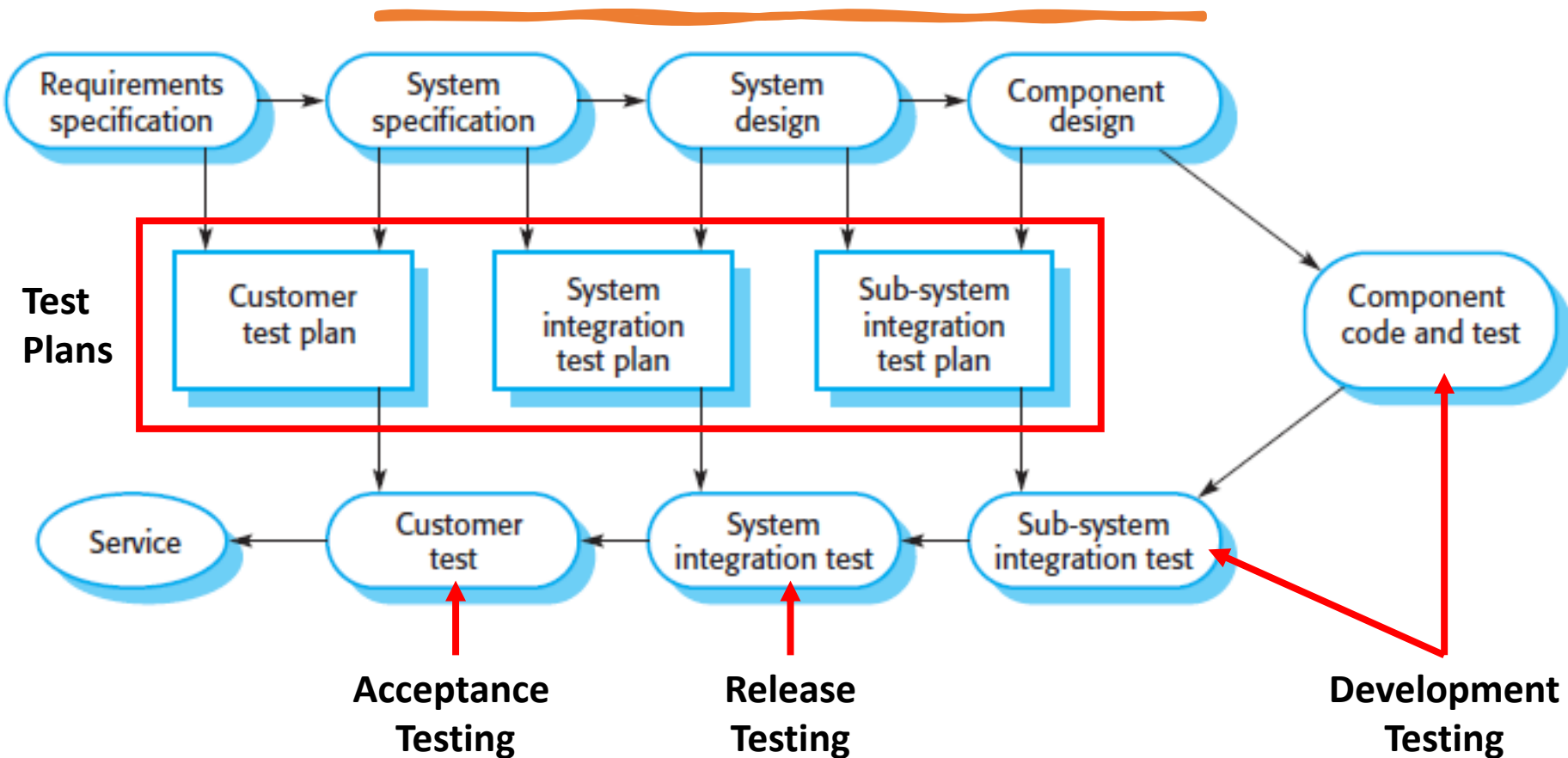
---

- Testing is driven by a set of test plans --- something often ignored!
- To determine if the development team has “finished” building something.
- Used for Test-Driven Development (TDD):

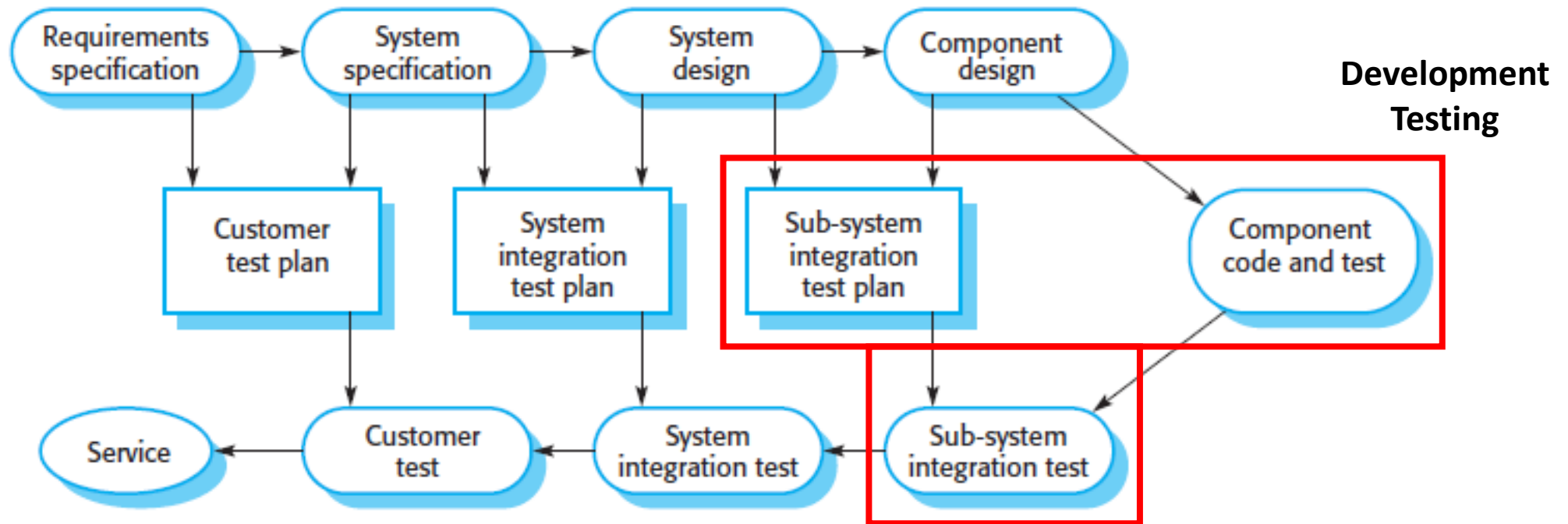
*“Testing is designed to show that the software **does what it is supposed to** and **discover program defects** before it is put to use.”*

- They are, therefore, part of the design phase.

# Testing Phase in a Plan-Driven Software Process (V-Model of Development)



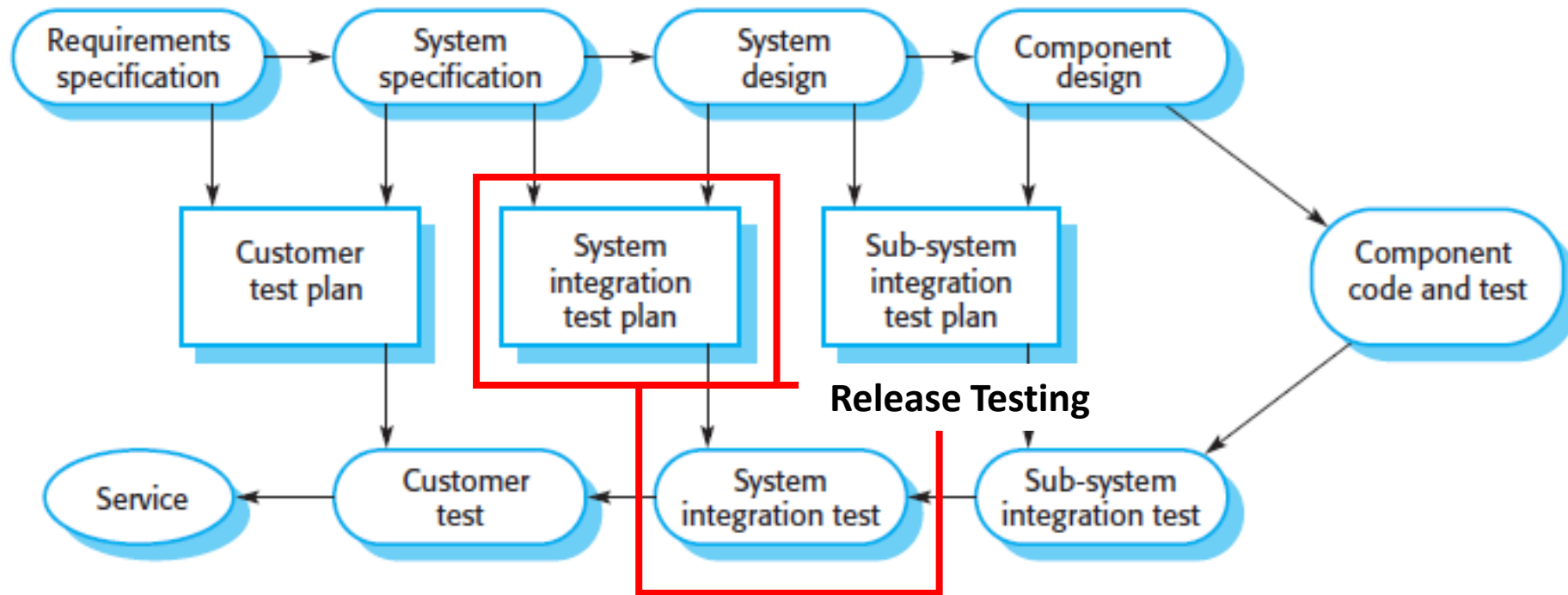
# Testing Phase in a Plan-Driven Software Process (V-Model of Development)



- **Development** Test Plans (**Component** Testing)
  - Let's prove that a class functions correctly.
  - Who will do it, with what data, on which platform?

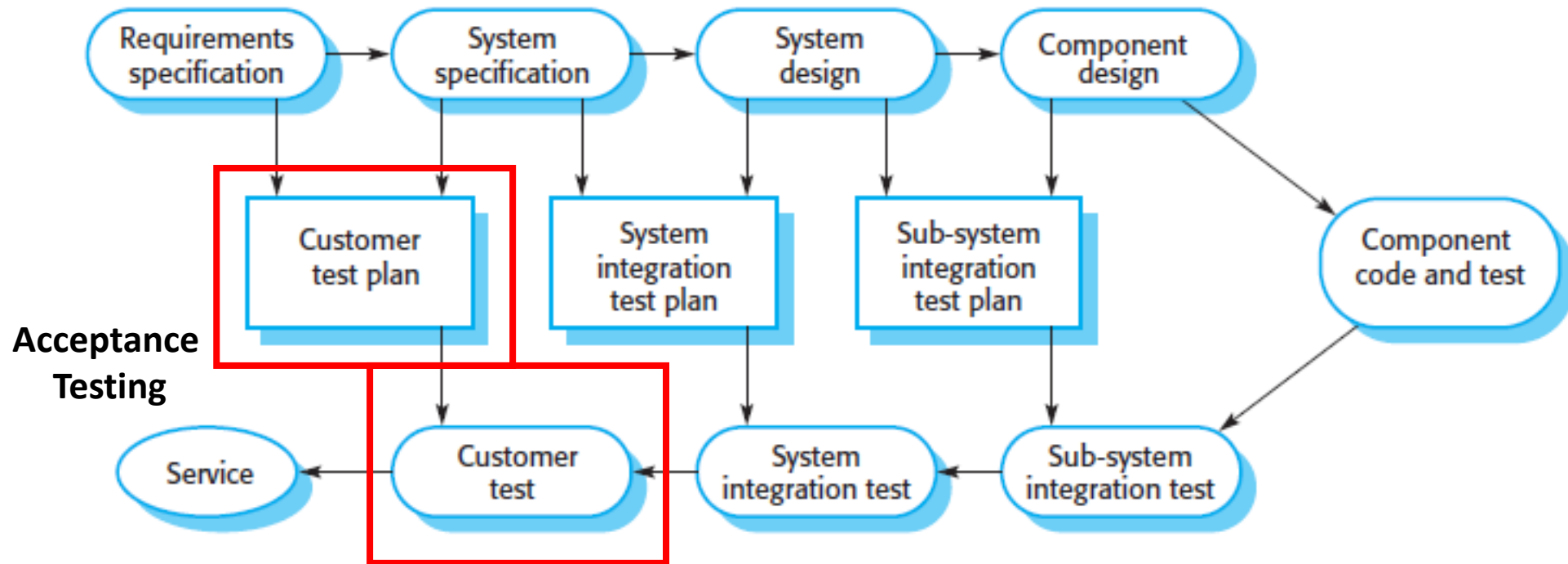


# Testing Phase in a Plan-Driven Software Process (V-Model of Development)



- **System/Integration** Test Plans (**System**/Integration Testing)
  - Let's prove that the software meets the specifications.
  - Tests if one class uses another class(es) correctly.
  - Or test that check that components interface correctly.

# Testing Phase in a Plan-Driven Software Process (V-Model of Development)



- **Acceptance** Test Plans (**Customer** Testing)
  - Shows that the software meets the requirements.
  - Often done with client - so they “accept” the software.

# Test Plans: What Are They Used For?

---

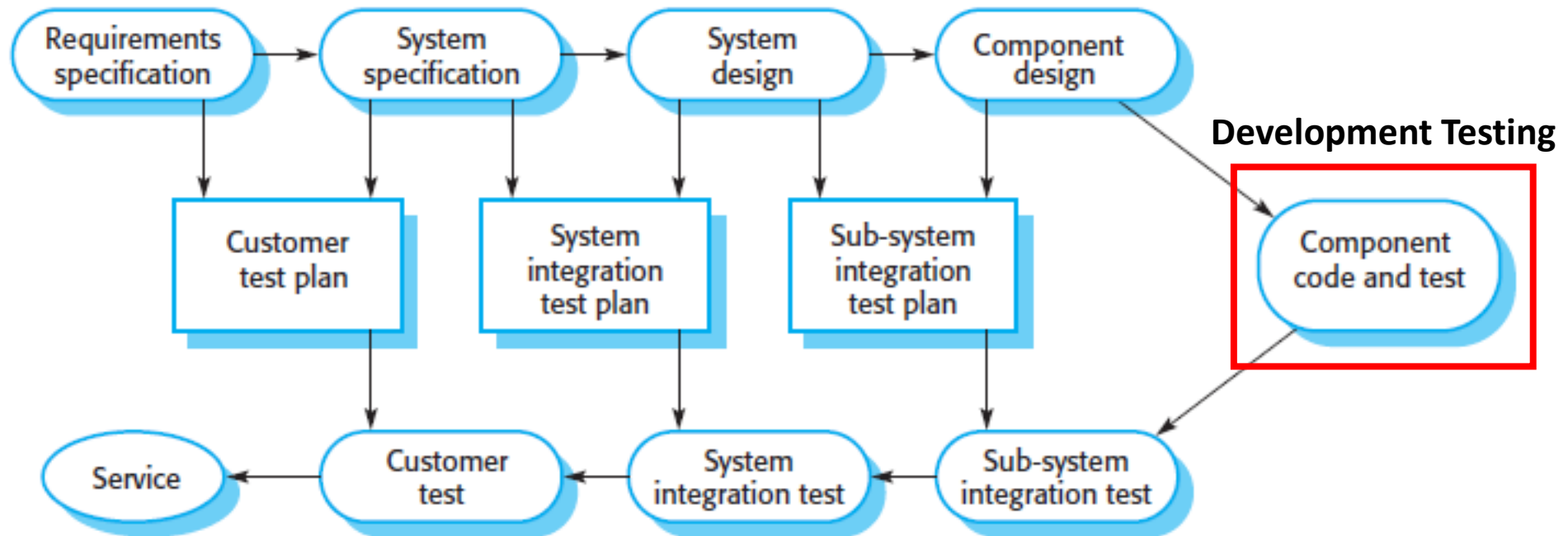
- To define what counts as “finished”:
  - How each stage back up the target is tested?
  - Where, when, by who etc.?
- 1) **Developers** use TPs to test code before delivering it.
- 2) **Managers** use TPs can estimate testing workload and schedule it – and include it in the budget.
- 3) Testing documentation serves as evidence to **clients** – of proper software engineering.

# Test Plans – The Overall Document

---

- Testing Process – description of the approach taken.
- Requirements Traceability – links between Requirements and Tests (can we trace?).
- Tested Items – list of things to be tested.
- Testing Schedule – schedule in relation to overall project.
- Test Recording Procedures – how test results will be recorded.
- Hardware/Software Requirements – for testing machines.
- Constraints – number of people, machines etc. needed.
- **System Tests** – a list of all the exact test cases that will be tested.

# Testing Phase in a Plan-Driven Software Process (V-Model of Development)



# Development Testing

---

- Unit Testing:
  - Individual program units or object classes are tested.
  - Focus on **testing functionality of objects or methods**.
- Component Testing:
  - Focus on **testing the component interfaces and composite components** (integration of several individual units).
- System Testing:
  - **Tested as a whole system**, where some or all the components in a system are integrated.



# Plan-Driven Development Testing Process

- Now, plan the test cases, and what data will test it.
- After built, run the the test, until it passes.

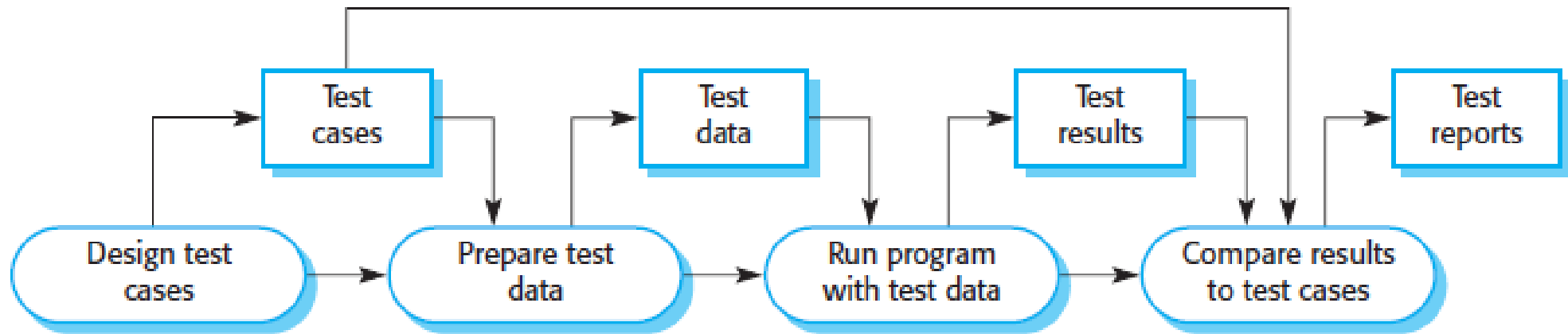


Fig: A Model of the Software Testing Process

# Plan-Driven Development Testing Process

- A statement of **what is being tested**.
- Specifications of the **inputs to the test** and the **expected output** from the system (test results).
- Specify the steps needed to carry out the test.
  - (if its not just 'run module with test data') - e.g. it might require other functions to be called first.

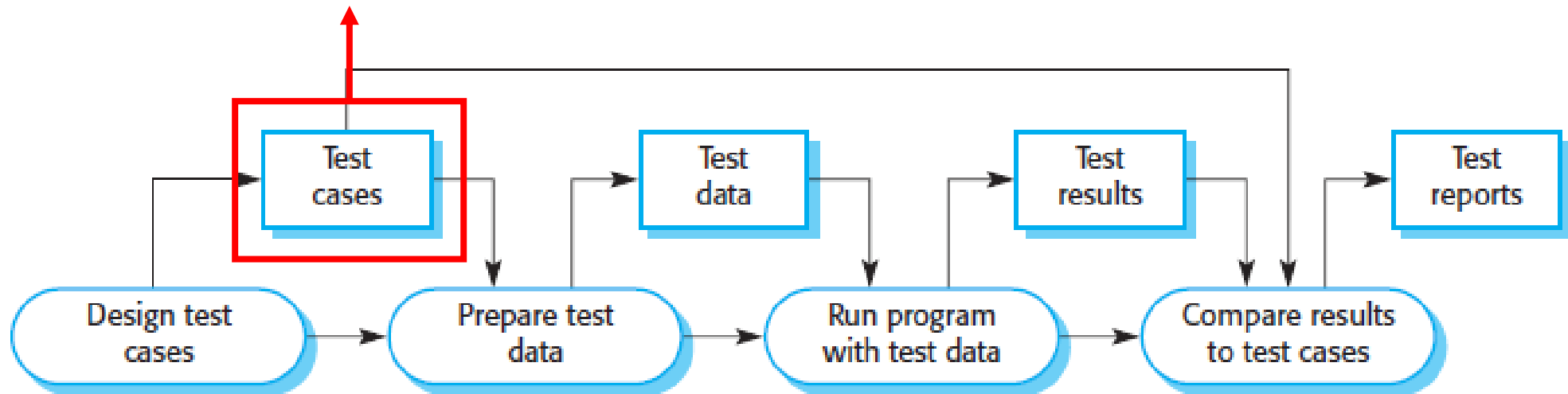


Fig: A Model of the Software Testing Process

# Plan-Driven Development Testing Process

- **Inputs** that have been **purposely planned to test a system**.
- Can be generated automatically.
- Automatic test case generation is impossible.
- Test execution can be automated.

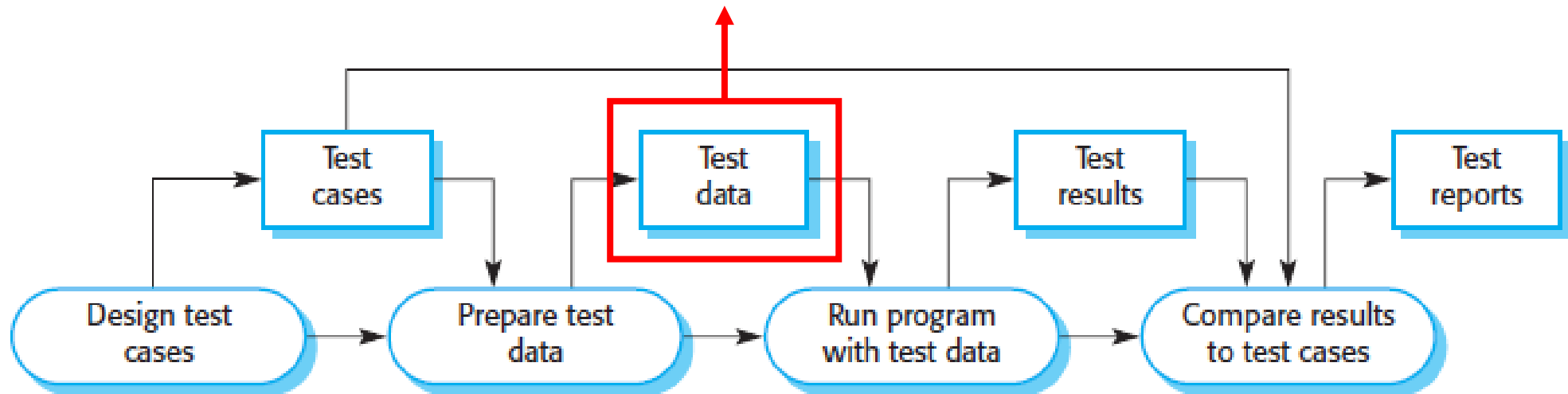
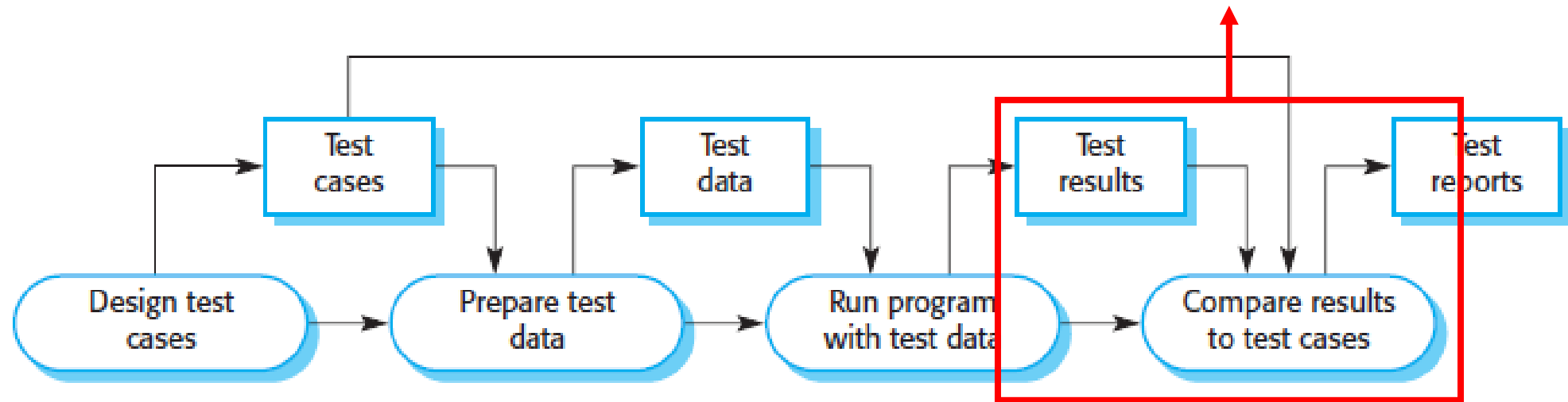


Fig: A Model of the Software Testing Process

- Test results are automatically compared with the predicted results.



### Fig: A Model of the Software Testing Process

# Test Planning



## Test planning

Test planning is concerned with scheduling and resourcing all of the activities in the testing process. It involves defining the testing process, taking into account the people and the time available. Usually, a test plan will be created that defines what is to be tested, the predicted testing schedule, and how tests will be recorded. For critical systems, the test plan may also include details of the tests to be run on the software.

<http://software-engineering-book.com/web/test-planning/>

# Test Plans

---

- Used to:
  - Define the scope of the testing effort.
  - Identify the features to be tested.
  - Identify the testing tasks to be performed.
  - Identify the personnel responsible for each task.
  - Schedule the testing effort.
  - Specify the resources required for the testing effort.
  - The risks and contingencies.
  - The pass/fail criteria.
  - The objectives of the testing.
  - The approach to be used.
  - The resources required.



# Test Plans – How to Write?

---

- A document that describes the scope, approach, resources, and schedule of testing activities for a particular project or release.
- Test plans are **tailored to the individual project**.
- Should be approved by the project manager before testing.

# Test Plans – Should Include?

---

- **Analyse** – product analysis.
- **Test Strategy** – designed to meet specific needs.
  - Should consider project's risk, available resources and schedule.
- **Test Objectives** – Define the goals and objectives of the testing process.
- **Test Criteria** – Mutual understanding what needs to be achieved and how it will be measured.
- **Resource Planning** – Ensure testing team has adequate resources such as test environment, test data, etc.
- **Schedule and Estimation** – Ensure all testing activities are completed promptly and efficiently.
- **Test Deliverables** – To assess if the agreed objectives are met.

# Test Plans – Types

---

## Validation Testing

Tests that show the software produces the right answer (when you give it all types of correct data).

## Defect Testing

Tests that show the software doesn't break (when you give it all types of incorrect data).

## Writing

Writing tests for validation only, is a common mistake.

# Test Plans


## >> Test Documents

---

Test ID	Reason	Input	Expected Output	Pass/Fail	Notes
1	A description of what we are testing	-1	error	Pass (date)	
2		4.99	converted to 5	Pass (date)	
3		1,000,000,000,000	error	1) FAIL (date) 2) Pass (date)	1) rounded to MAXINT ACTION: check for x>MAXINT
4	The next thing we are testing	-1	error	Pass (date)	

# Test Plans >> Test Documents

- [https://www.softwaretestinghelp.com/wp-content/qa/uploads/2014/02/Live\\_Project\\_Test\\_Plan\\_SoftwareTestingHelp.pdf](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2014/02/Live_Project_Test_Plan_SoftwareTestingHelp.pdf)

<b>Project Name:</b>	Google Email	 <a href="http://www.SoftwareTestingMaterial.com">www.SoftwareTestingMaterial.com</a>
<b>Module Name:</b>	Login	
<b>Reference Document:</b>	If any	
<b>Created by:</b>	Rajkumar	
<b>Date of creation:</b>	DD-MMM-YY	
<b>Date of review:</b>	DD-MMM-YY	

TEST CASE ID	TEST SCENARIO	TEST CASE	PRE-CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT	POST CONDITION	ACTUAL RESULT	STATUS (PASS/FAIL)
TC_LOGIN_001	Verify the login of Gmail	Enter <b>valid User Name</b> and <b>valid Password</b>	1. Need a valid Gmail Account to do login	1. Enter User Name 2. Enter Password 3. Click "Login" button	<Valid User Name> <Valid Password>	Successful login	Gmail inbox is shown		
TC_LOGIN_001	Verify the login of Gmail	Enter <b>valid User Name</b> and <b>invalid Password</b>	1. Need a valid Gmail Account to do login	1. Enter User Name 2. Enter Password 3. Click "Login" button	<Valid User Name> <Invalid Password>	A message "The email and password you entered don't match" is shown			
TC_LOGIN_001	Verify the login of Gmail	Enter <b>invalid User Name</b> and <b>valid Password</b>	1. Need a valid Gmail Account to do login	1. Enter User Name 2. Enter Password 3. Click "Login" button	<Invalid User Name> <Valid Password>	A message "The email and password you entered don't match" is shown			
TC_LOGIN_001	Verify the login of Gmail	Enter <b>invalid User Name</b> and <b>invalid Password</b>	1. Need a valid Gmail Account to do login	1. Enter User Name 2. Enter Password 3. Click "Login" button	<Invalid User Name> <Invalid Password>	A message "The email and password you entered don't match" is shown			

# Test Case Document – Example

source: <https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2017/10/Test-Case-Example1.jpg>

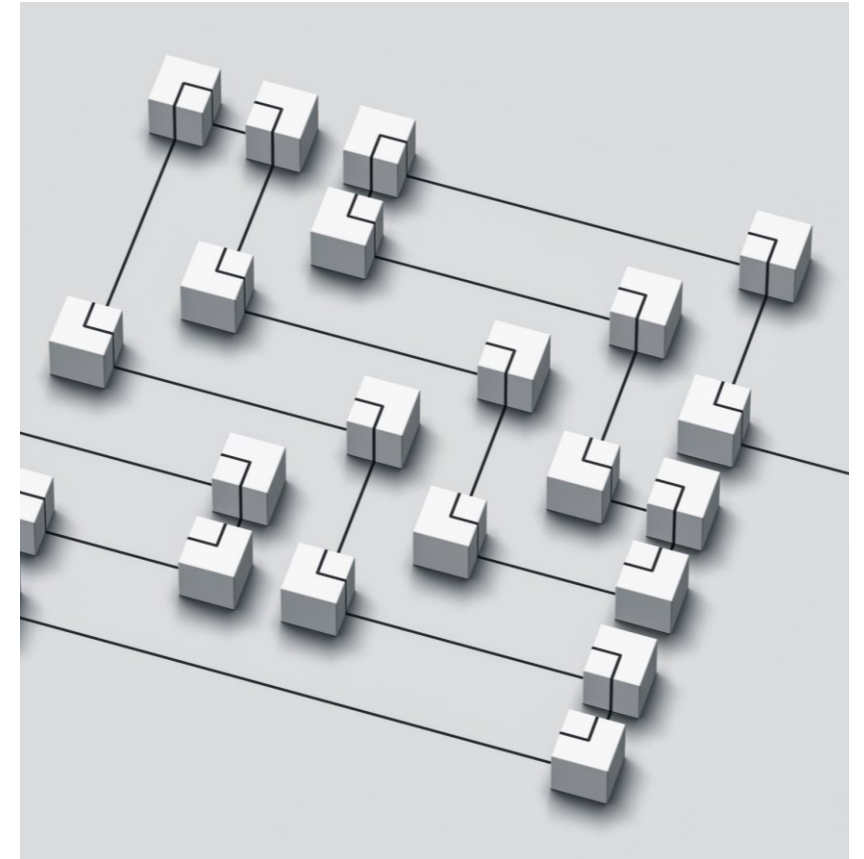
Test Scenario ID	Login-1		Test Case ID	Login-1A			
Test Case Description	Login – Positive test case		Test Priority	High			
Pre-Requisite	A valid user account		Post-Requisite	NA			
Test Execution Steps:							
S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Launch application	https://www.facebook.com/	Facebook home	Facebook home	IE-11	Pass	[Priya 10/17/2017 11:44 AM]: Launch successful
2	Enter correct Email & Password and hit login button	Email id : test@xyz.com Password: *****	Login success	Login success	IE-11	Pass	[Priya 10/17/2017 11:45 AM]: Login successful



# Summary

---

1. Low-level designs (object-oriented design) are there to guide developers.
2. These are the final outputs of the system specifications phase.
3. Planning for testing is part of specifications phase.
  - Development Testing – Unit/Component Test Plan
  - System/Integration Testing – Integration Test Plan
  - Acceptance Testing – Customer Test Plan
4. Plan-driven development testing process.
  - Test cases – test data – test results – test report.



# Optional Materials

---

- Google Chrome Test Automation Lab
  - <https://www.youtube.com/watch?v=08CyrK2d1t0&list=PLf2uDdNGIEWFUBspTli433u2Ap0ttxN6L&index=14>
- Martin Fowler: Software Design in the 21<sup>st</sup> century.
  - <https://www.youtube.com/watch?v=6wDooptEqk>
- A Refactoring Book
  - <https://www.refactoring.com/>
- Is TDD (Test Driven Development) dead? Of course not!
  - <https://www.youtube.com/watch?v=PCEHRFHKZSk>

