Software Engineering COMP1035

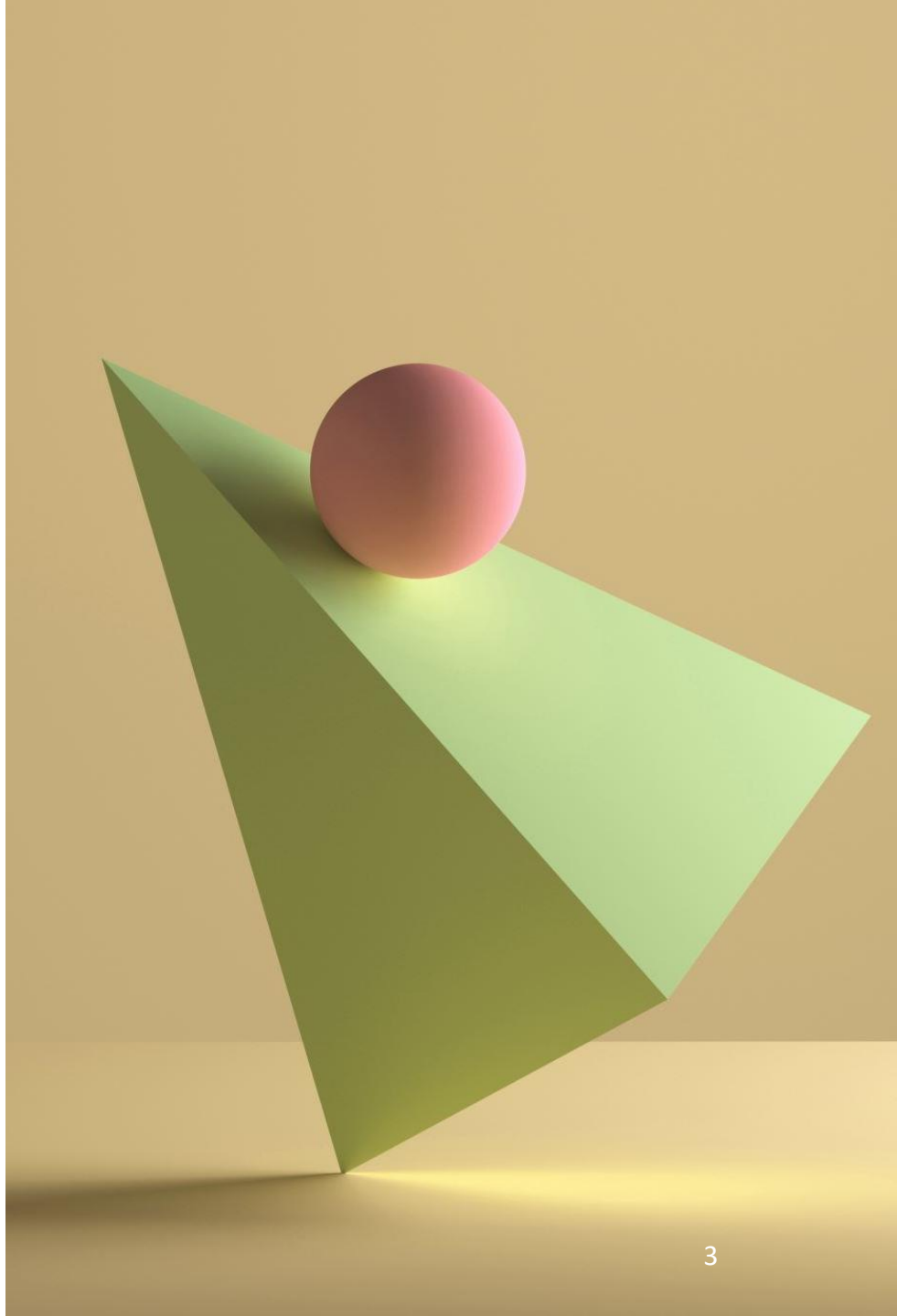
Lecture 09

OO Design & Test Plans

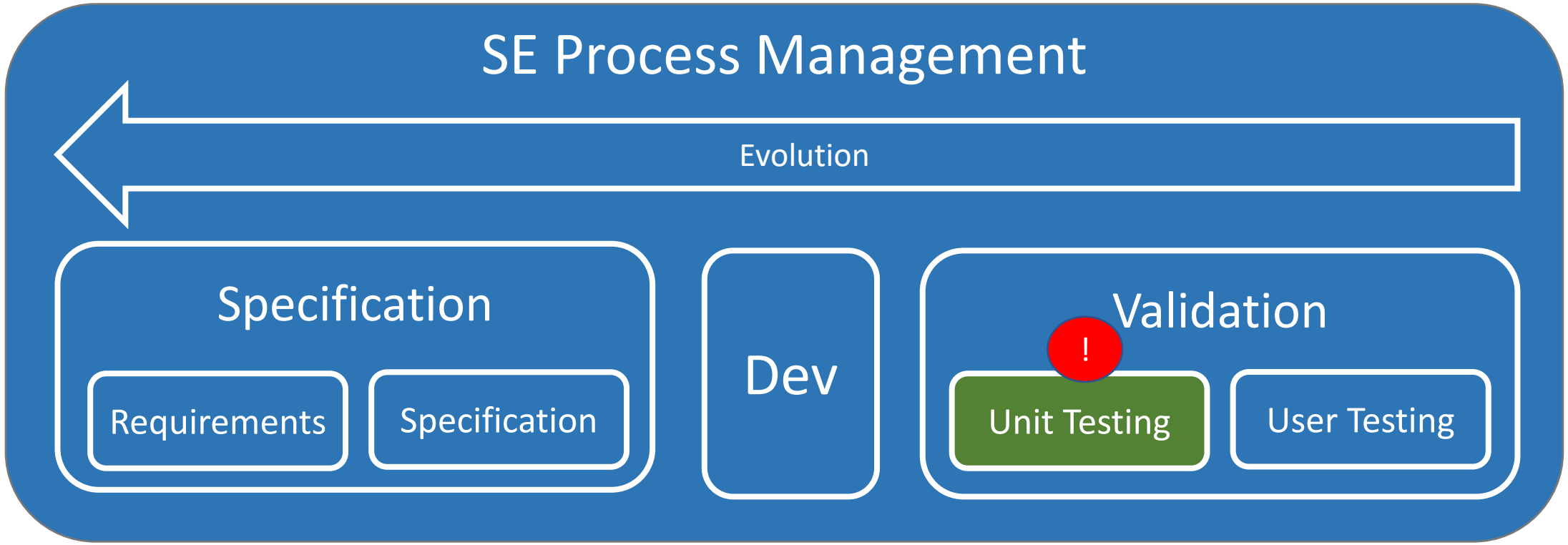
Today’s Objectives:

1. Low-level designs are there to guide developers.
2. These are the final outputs of the Specifying phase.
3. Planning testing is part specifying phase.

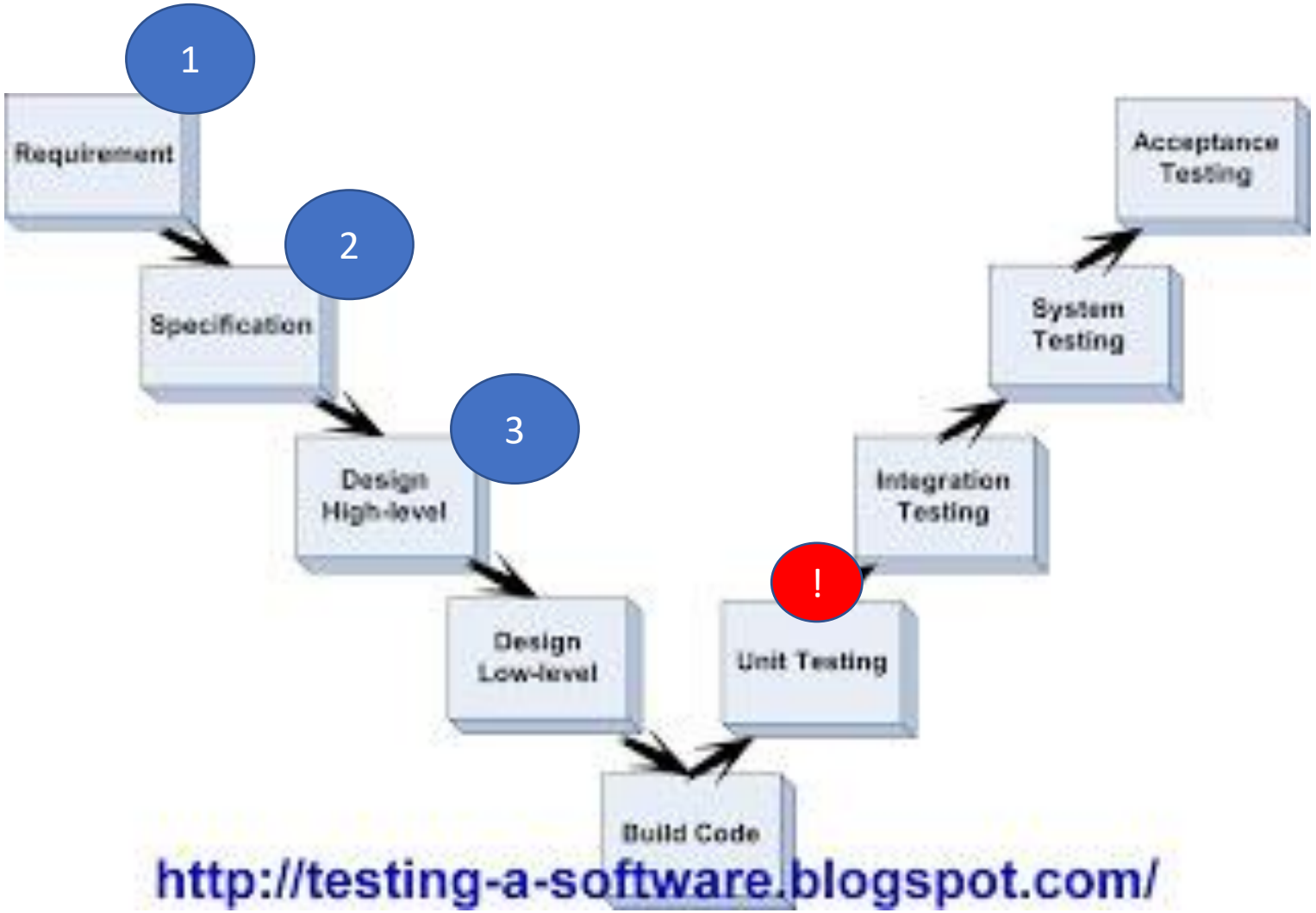
Where We Are In the Process



Keeping Track of SE Module



Keeping Track of SE Module



Where Are We?

- We want to have enough details to give to the development team.
  - “Here, you build this”.
  - Not - here, please design this software.

- Creating a final low-level design is often led by a Solution Architect
  - “Perhaps the most important tool in the toolbox is a visual documentation language, such as UML.”
  - “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.

From the End of Last Lecture

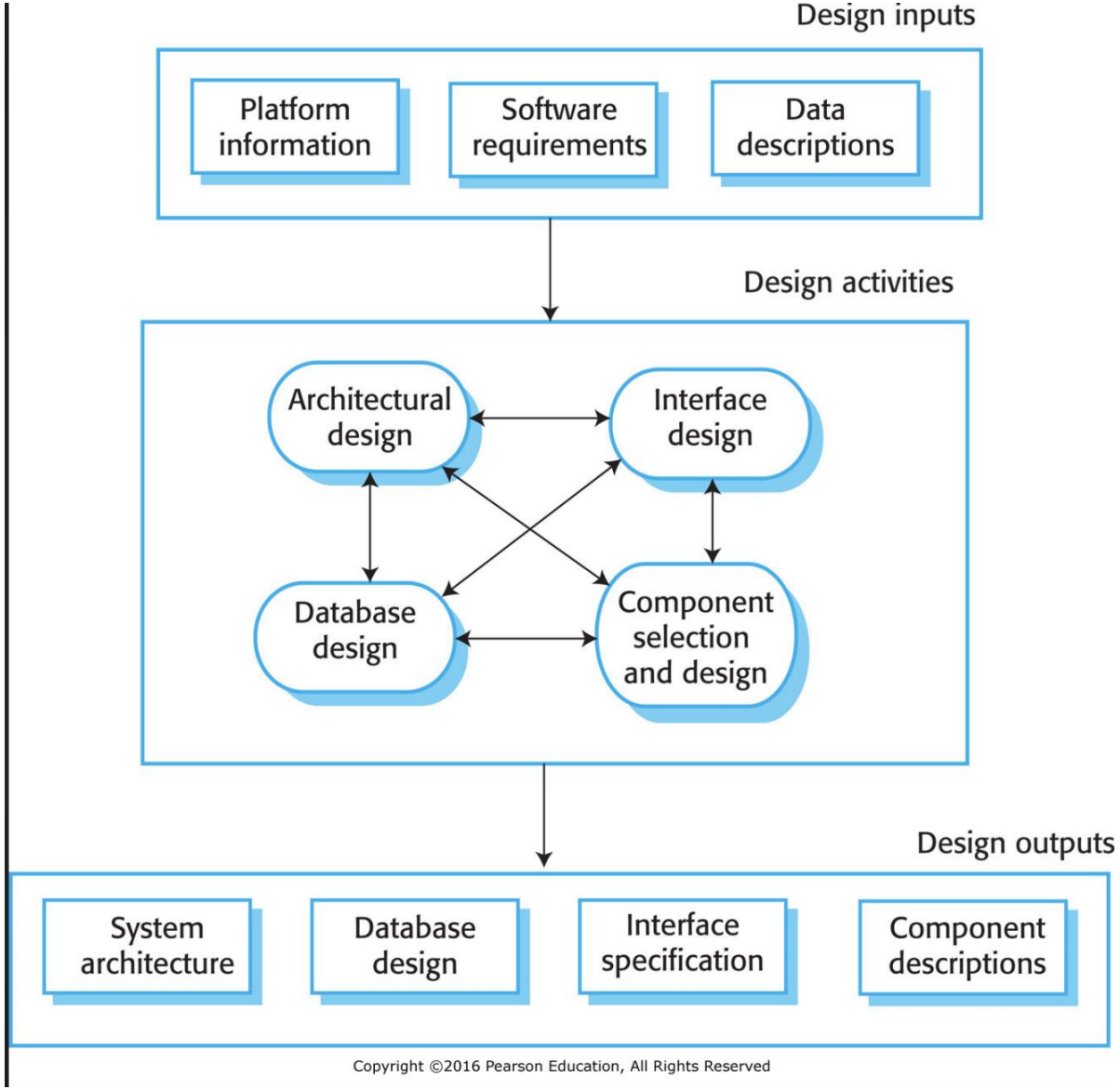


Fig. A general model of the design process.

Some Notes on Final Spec Outputs

- **Architecture** Design
  - Now we are talking about the internal architecture of the objects, classes, use of APIs.
  - Like **MVC** (Model View Controller) architecture (not covered in this module).
- **Interface** Design
  - Not just the ‘user’ interface, also any interfaces with other services (that might be developed by different teams).
  - E.g., the web team, the mobile dev team, the server-side team.
- **Database** Design
  - Developers will presume that they can request from the DB in their code, as specified.
  - Other types of DB, however, could be specified.
- **Component** Design
  - The exact class structure inside each component.

Low Level Object-Oriented Design

How To Conduct Objected-Oriented Design

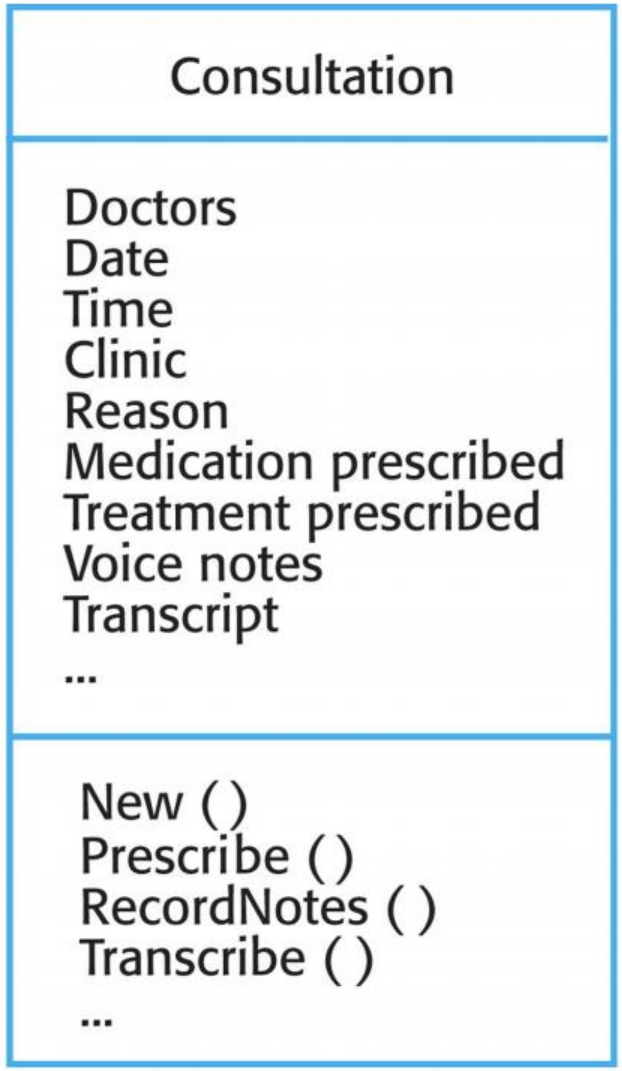
- Understand and define the context and the external interactions with the system.
- Design the system architecture.
- Identify the principal objects in the system
- Develop design models.
- Specify interfaces.

Component Design In UML

- Presuming Object-Oriented (in this module)
  - Classes that represent objects
  - Objects store data
  - Objects have responsibilities

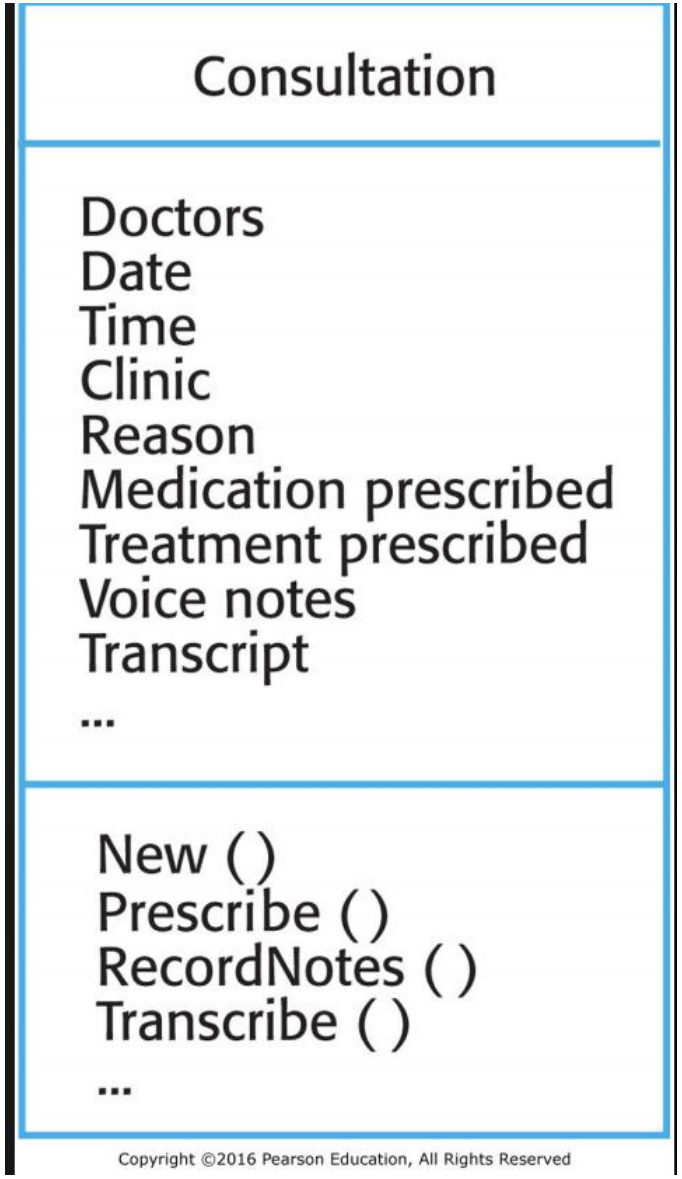
Class Diagram

Figure 5.10 A Consultation class



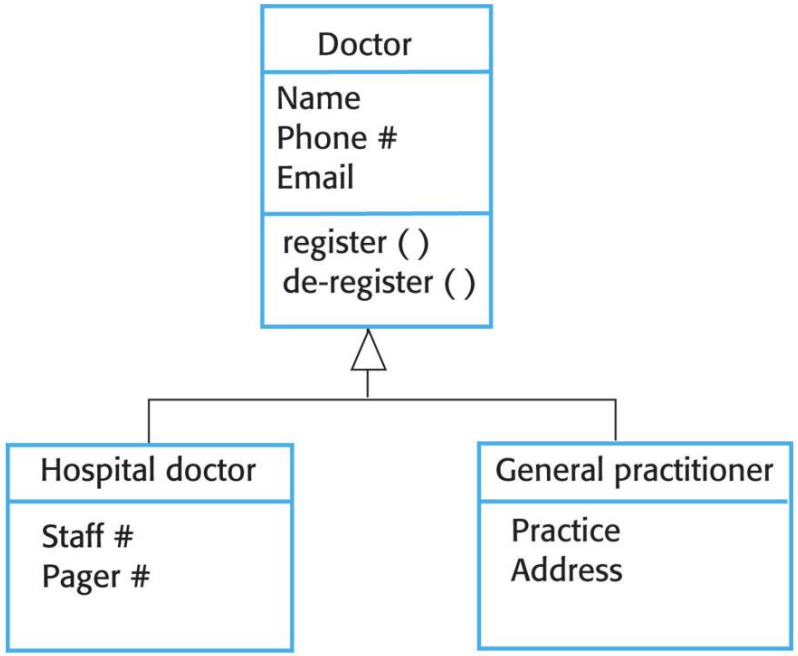
Class Description Documents

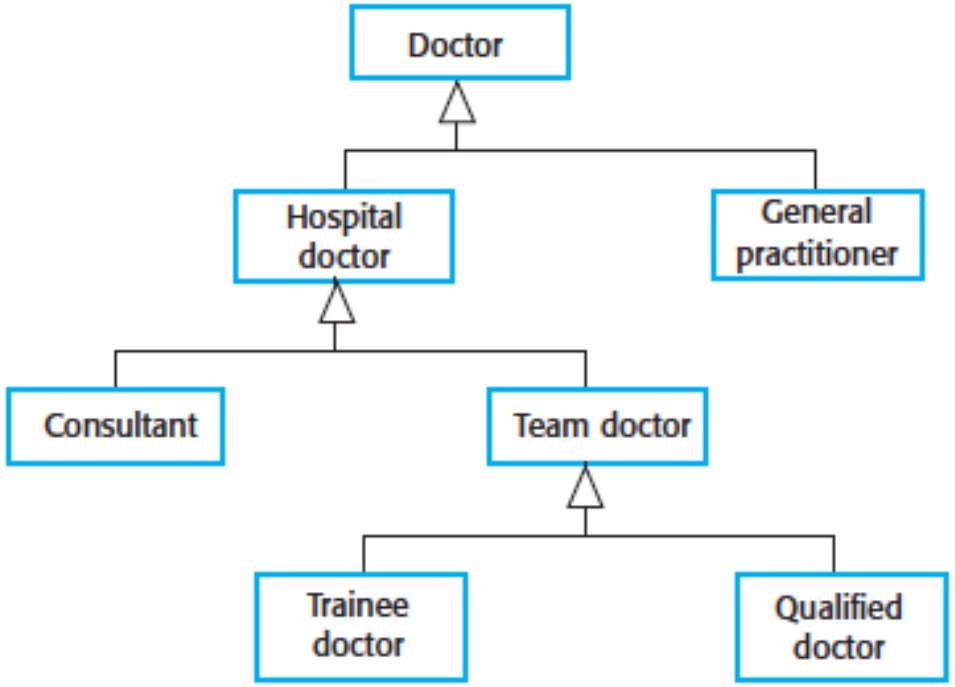
- Adding detail to Classes.
- Both guides developers and acts as first version of documentation.
  - For each variable:
    - What data format?
    - What will it be used for?
  - For each method/function:
    - What information does it receive (and why)?
    - What does it do?
    - What output does it produce (and why)?
    - Who will use the output?
  - Finally, how to know it works as expected?



Class Diagram

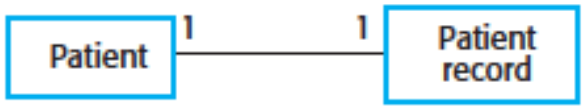
Figure 5.12 A generalization hierarchy with added detail





Defining Classes And Their Relationships

Figure 5.8 UML Classes and association



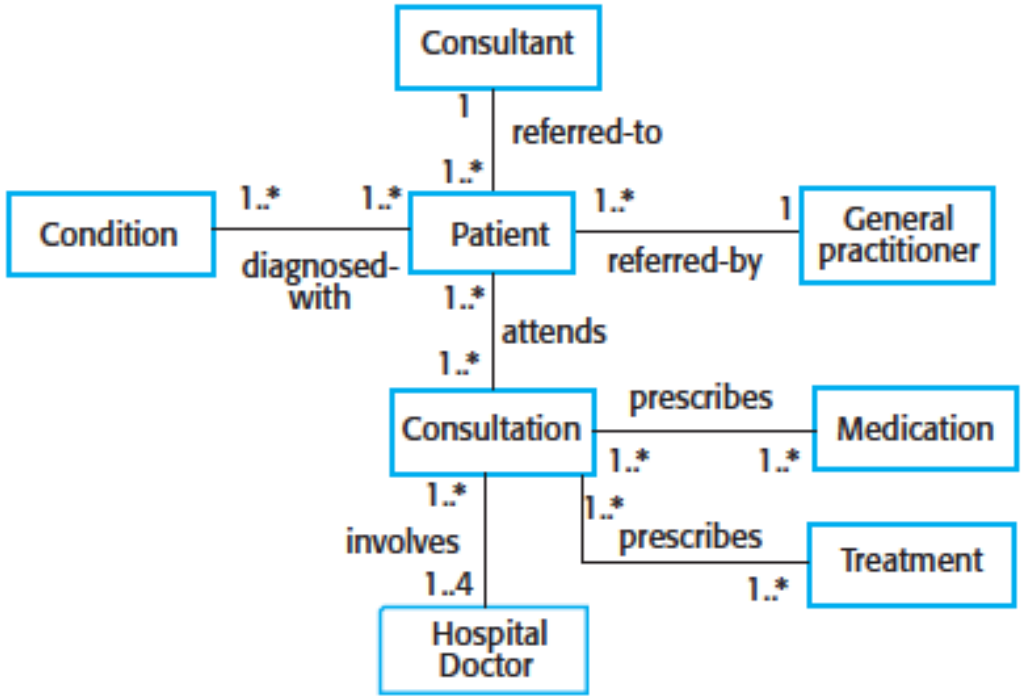


Figure 5.9 Classes and associations in the Mentcare system

Entity-Relationship Diagram (ERD)

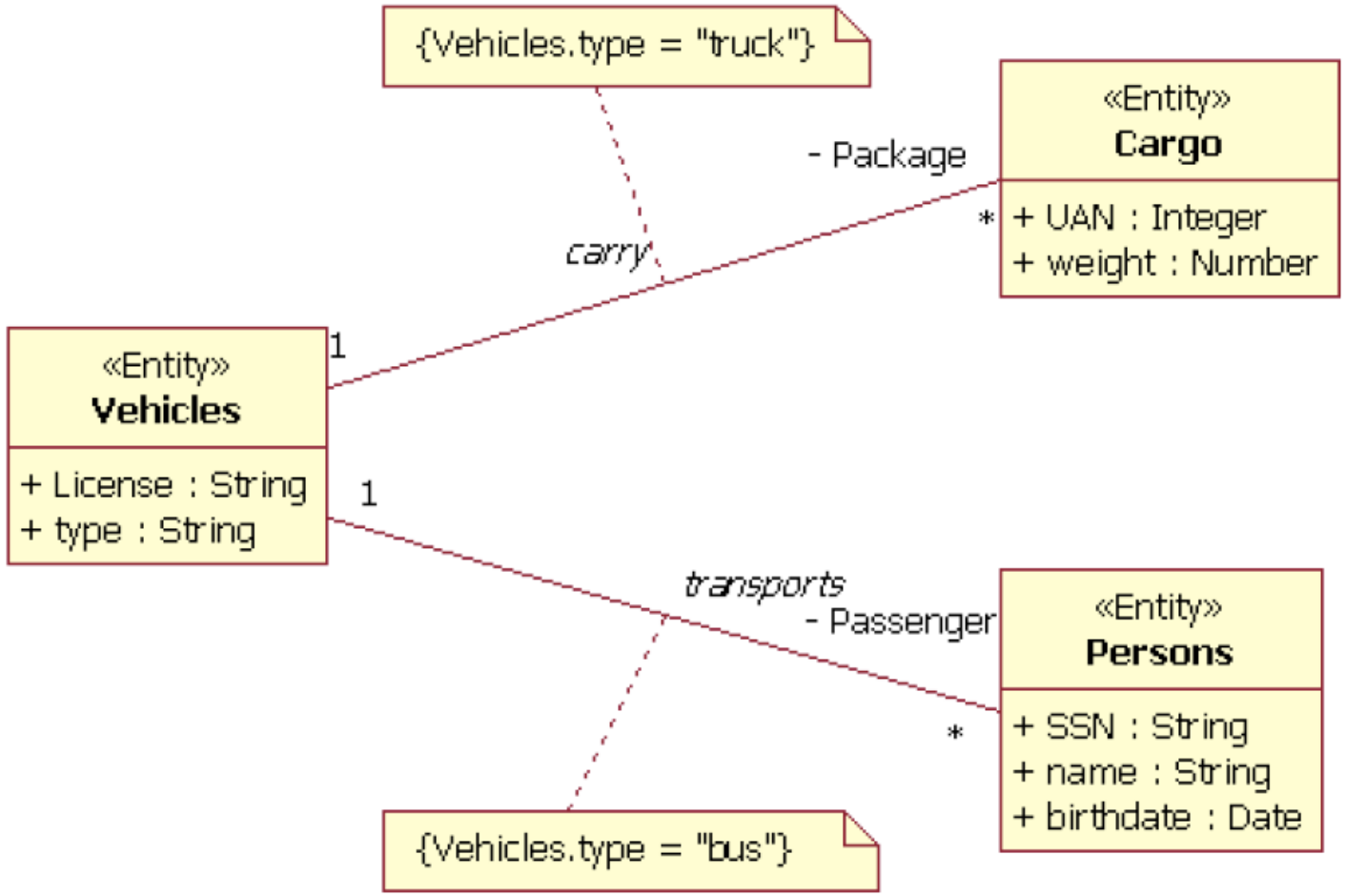


Figure 19 Categorization of entity types defines the criteria for relationship types – Cargo is important for Vehicles of the type "truck"; Persons can be transported for the Vehicles of the type "bus"

Figure source: https://www.ibm.com/developerworks/rational/library/content/03July/2500/2785/2785_uml.pdf

Test Plans

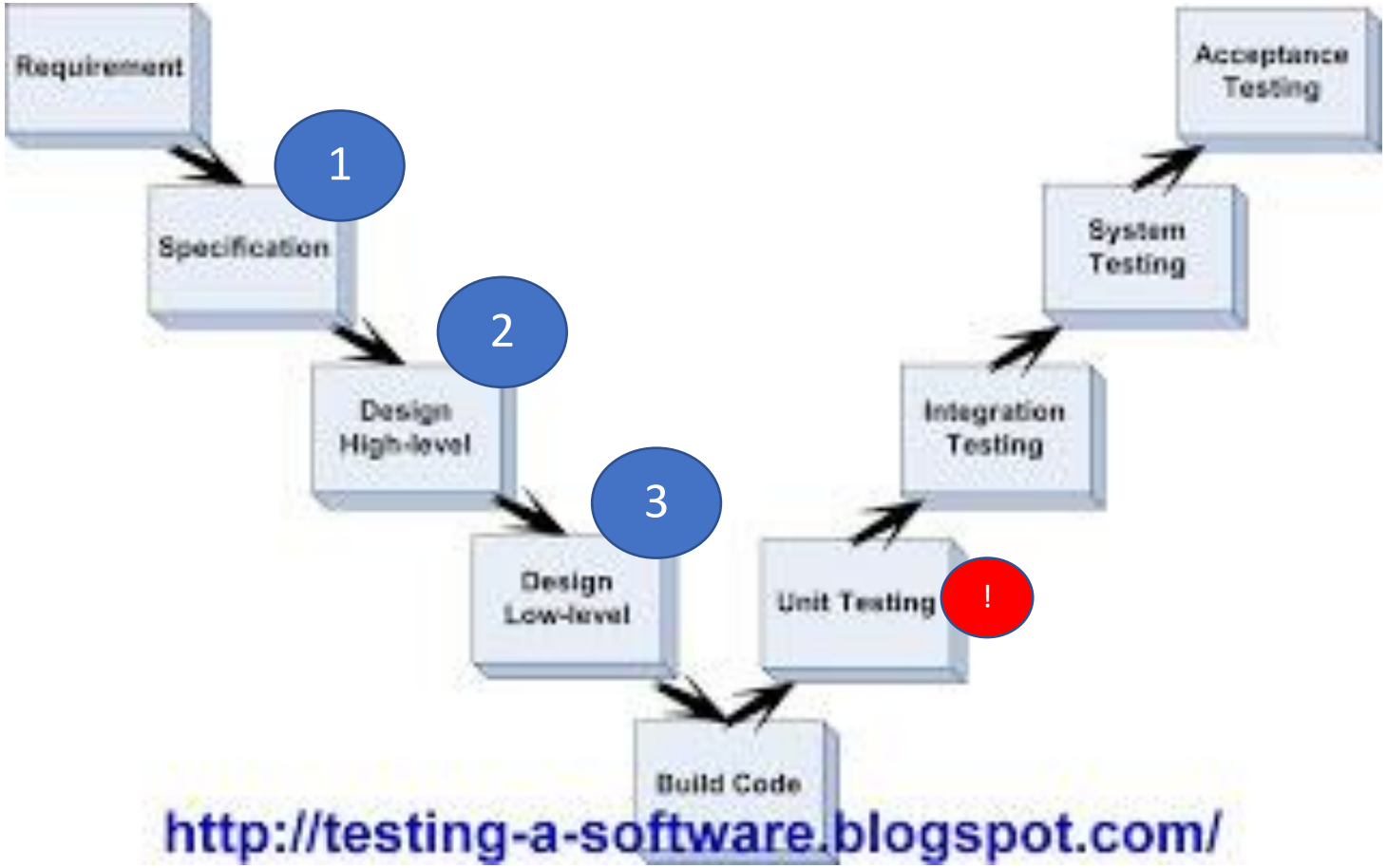
Test Plans

- Something often ignored.
- Determines when a programmer has ‘finished’ building something.
- Used for Test Driven Development (future class):

“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.

Keeping Track of SE Module



Testing Phase In Plan Driven Approach

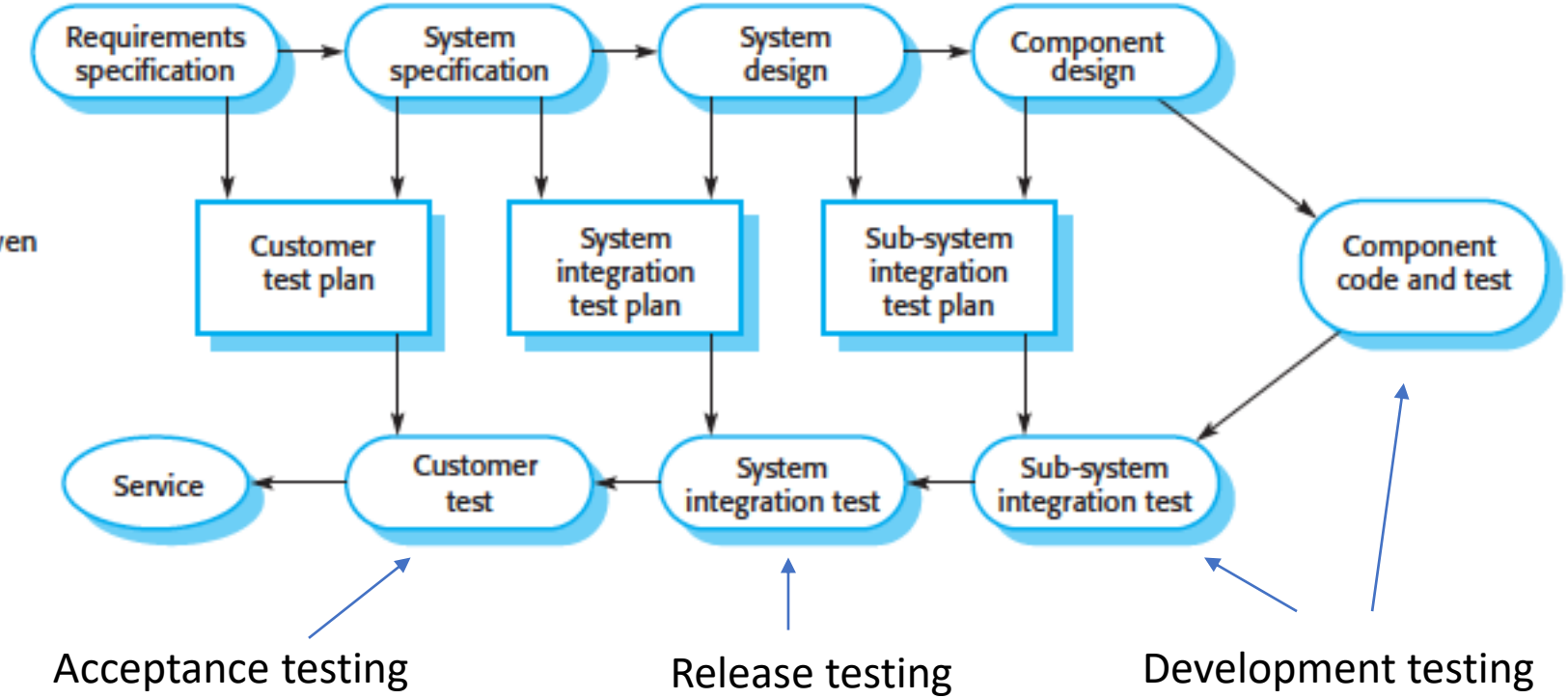


Figure 2.7 Testing phases in a plan-driven software process

Test Plans

- **Development** Test Plans (Unit Testing)
  - Let's prove that a class functions correctly.
  - Who will do it, with what data, on which platform?
- **System/Integration** Test Plans (System/Integration Testing)
  - Let's prove that the software meets the Specs.
  - Tests if one class uses another class(es) correctly.
  - Or test that check that components interface correctly.
- **Acceptance** Test Plans (Acceptance 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.

- 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 Plan Driven Approach

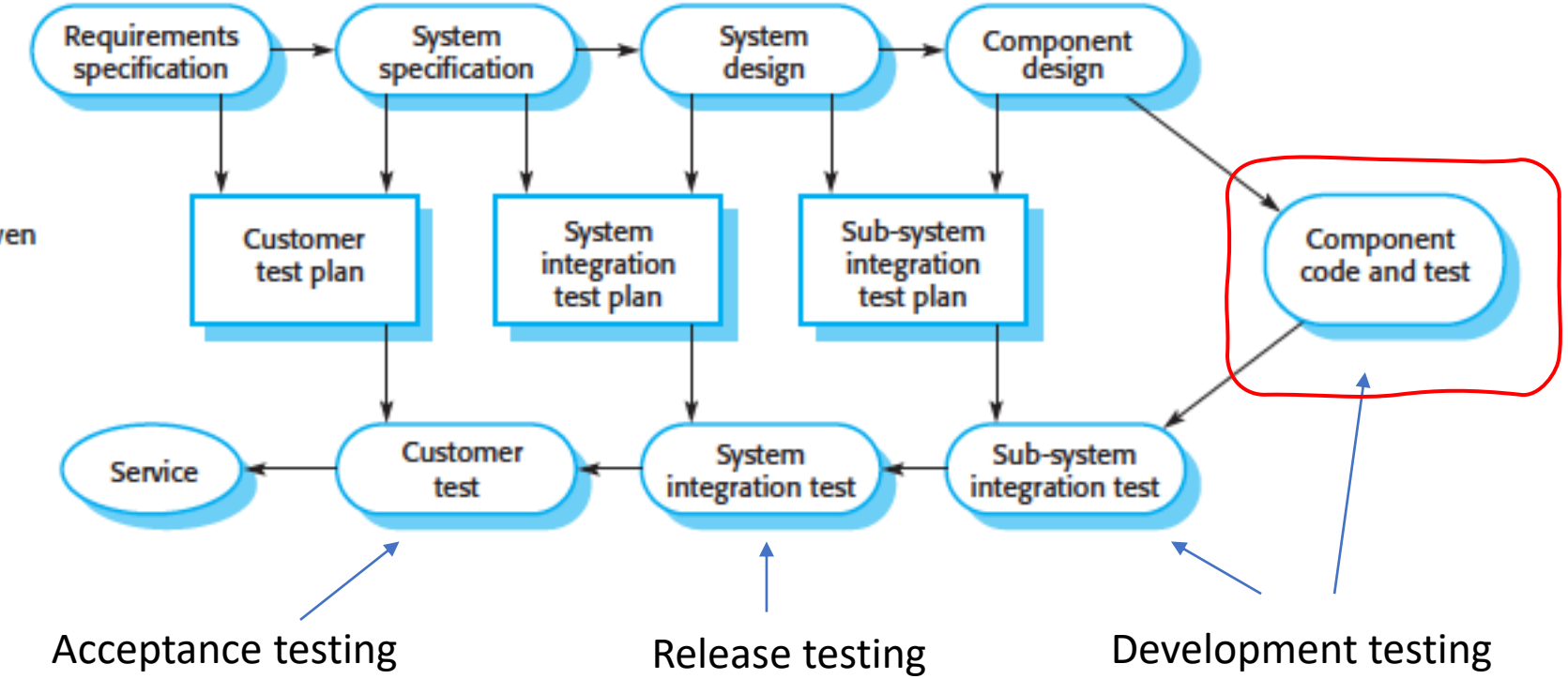
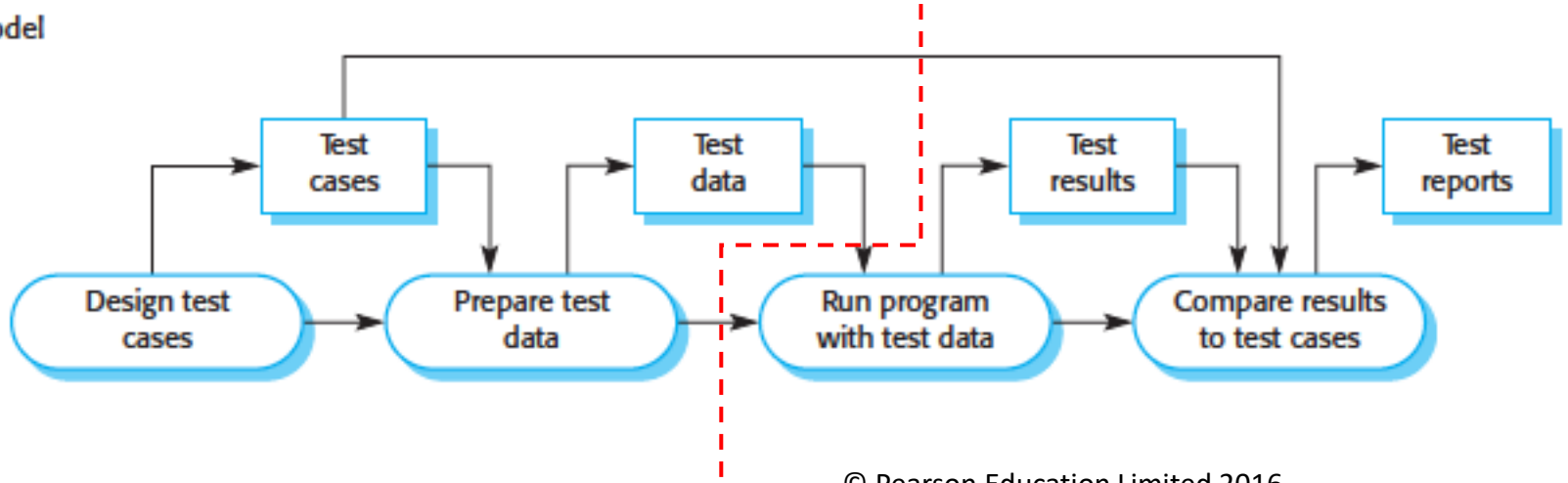


Figure 2.7 Testing phases in a plan-driven software process

Designing Test Cases

Figure 8.3 A model of the software testing process



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

About Each Test Case

- A statement of what is being tested.
- Specifications of the inputs to the test.
- What is the expected output from the system?
- 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.

Test Plans - Types

Testing	Testing	Writing
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 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

Project Name:	**Google Email**
Module Name:	**Login**
Reference Document:	**If any**
Created by:	**Rajkumar**
Date of creation:	**DD-MMM-YY**
Date of review:	**DD-MMM-YY**

STM

www.SoftwareTestingMaterial.com

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 valid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Valid User Name>	Successful login	Gmail inbox is shown		
				2. Enter Password	<Valid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter valid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Valid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Invalid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter invalid User Name and valid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Invalid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Valid Password>				
				3. Click "Login" button					
TC_LOGIN_001	Verify the login of Gmail	Enter invalid User Name and invalid Password	1. Need a valid Gmail Account to do login	1. Enter User Name	<Invalid User Name>	A message "The email and password you entered don't match" is shown			
				2. Enter Password	<Invalid Password>				
				3. Click "Login" button					

Test Case Document – Example

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

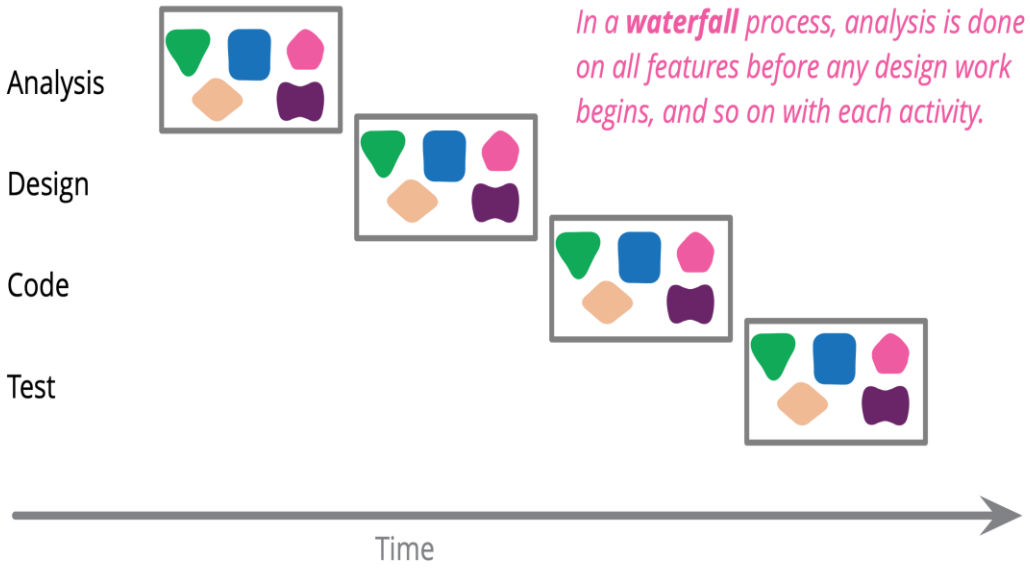
Test Execution Steps:

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

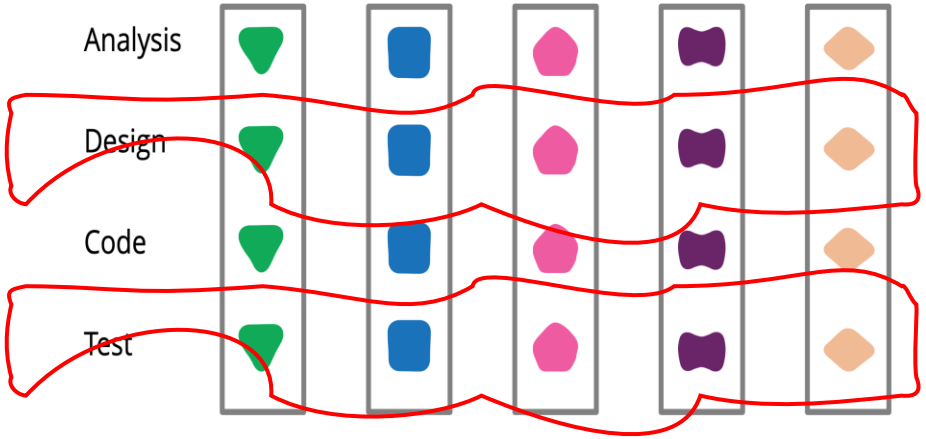
Figure source: https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2017/10/Test-Case-Example1.jpg

Essential Takeaway From Real-World Practices

- Waterfall is not dead.
- But modern software industry massively adopted iterative approach.



*In a **waterfall** process, analysis is done on all features before any design work begins, and so on with each activity.*



*In an **iterative** process, all activities are done for one feature before work begins on another feature.*

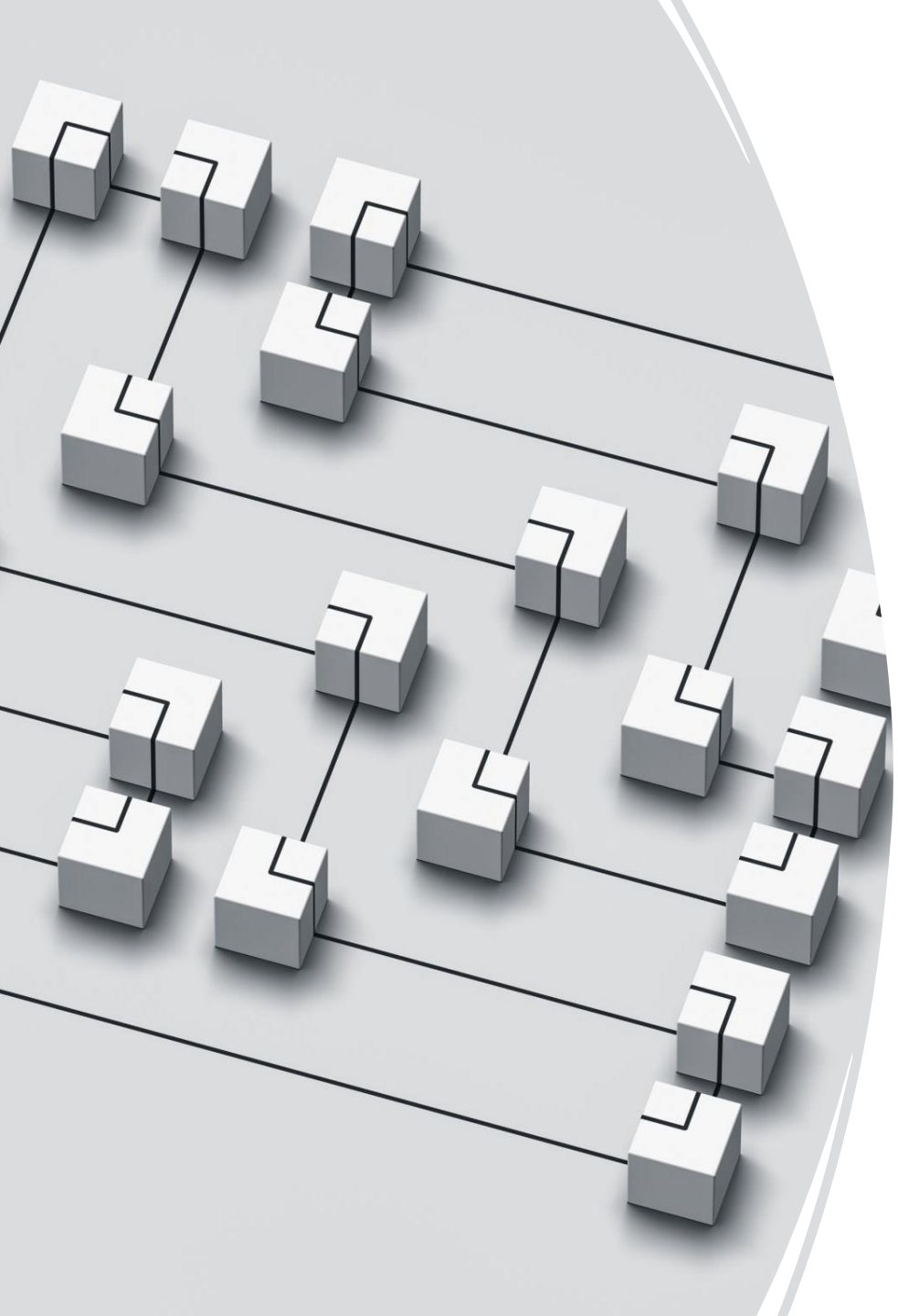
Optional Materials

- Google Chrome Test Automation Lab
  - https://www.youtube.com/watch?v=08CyrK2d1t0&list=PLf2uDdNGIEWFUBspTli433u2Ap0ttxN6L&index=14

- Martin Fowler: Software Design in the 21st century.
  - https://www.youtube.com/watch?v=6wDoopbtEqk

- A Refactoring Book
  - (https://www.refactoring.com/)

- Is TDD (Test Driven Development) dead? Of course not!
  - (https://www.youtube.com/watch?v=PCEHRFHKZSk)



Summary

1. Low-level designs (Object Oriented Design) are there to guide developers.
2. These are the final outputs of the Specifying phase.
3. Planning testing is part specifying phase.
   - Type of testing:
     - Unit Testing, Integration Testing, System Testing, and Acceptance Testing.


THANK
YOU