Тестування програмного забезпечення

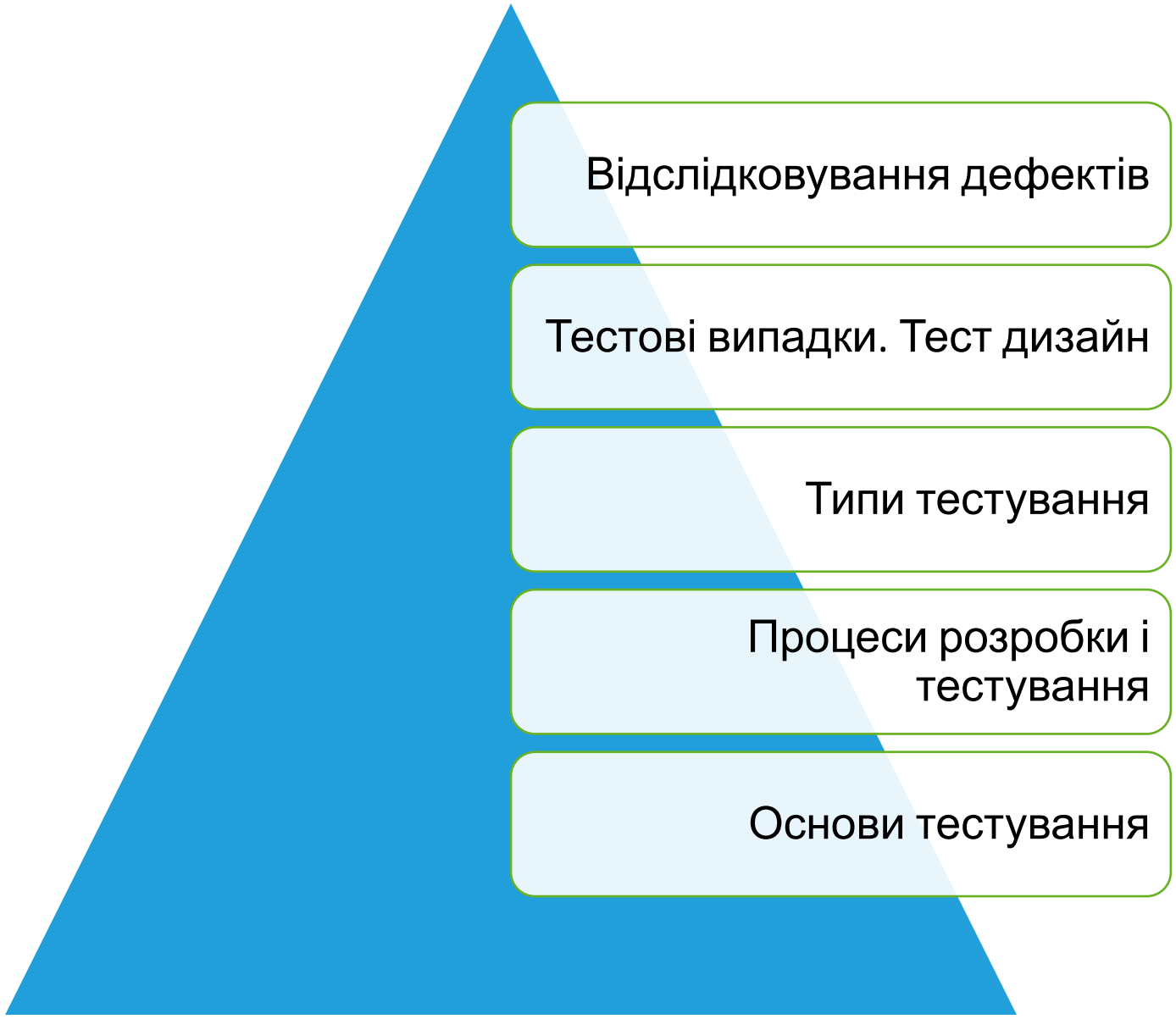
Prometheus online course

Тиждень 1

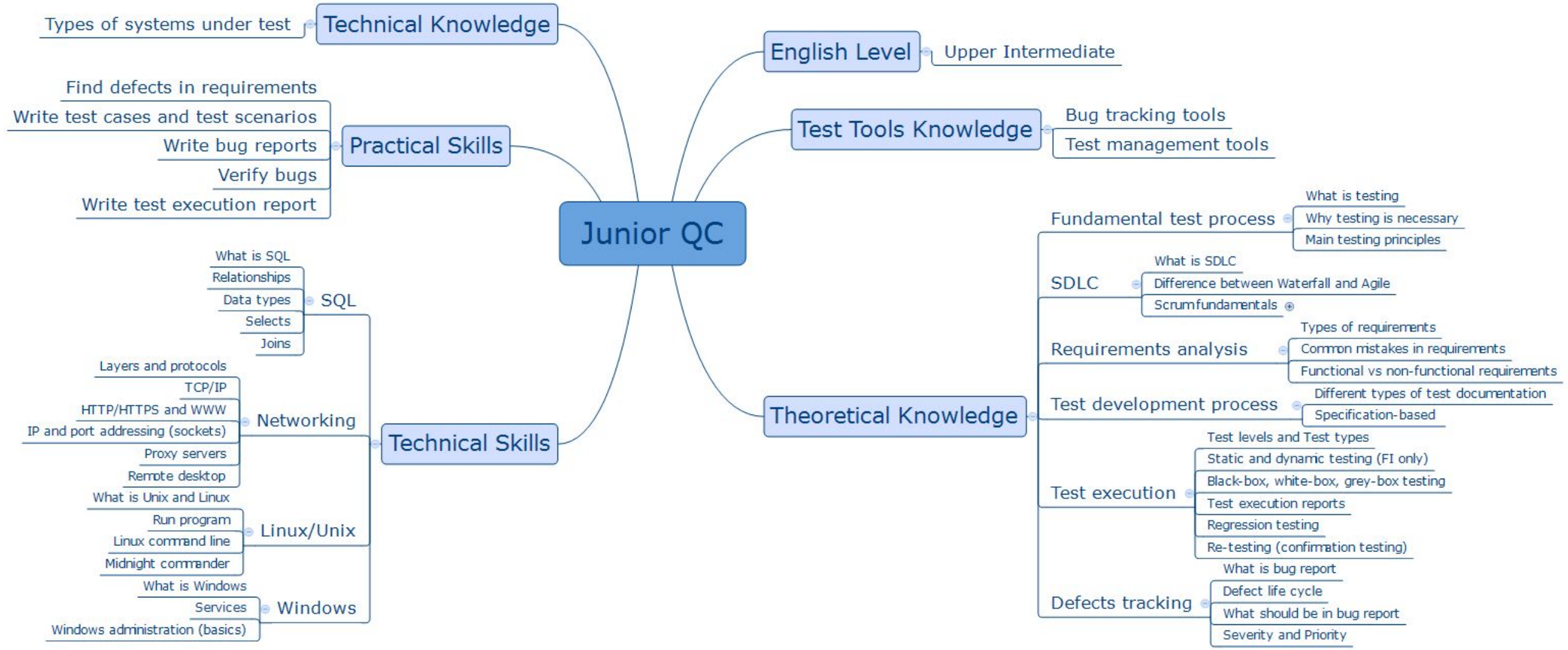
Тиждень 1

1. Вступ. Структура курсу
2. Якість та тестування
3. Основні концепції тестування

Тиждень 1. Вступ. Структура курсу



Тиждень 1. Необхідні знання



Тиждень 1. Якість та тестування

Якість – сукупність характеристик об’єкта, що дозволяє задовольняти встановлені потреби.

Якістю є міра, в якій продукт відповідає вимогам користувача, на початку свого існування (ISO 9000)

Якість – ніколи не буває випадковою. Вона є результатом високої мети, максимальних зусиль, правильного напрямку і майстерного виконання. Вона являє собою розумний вибір багатьох альтернатив. William A. Foster

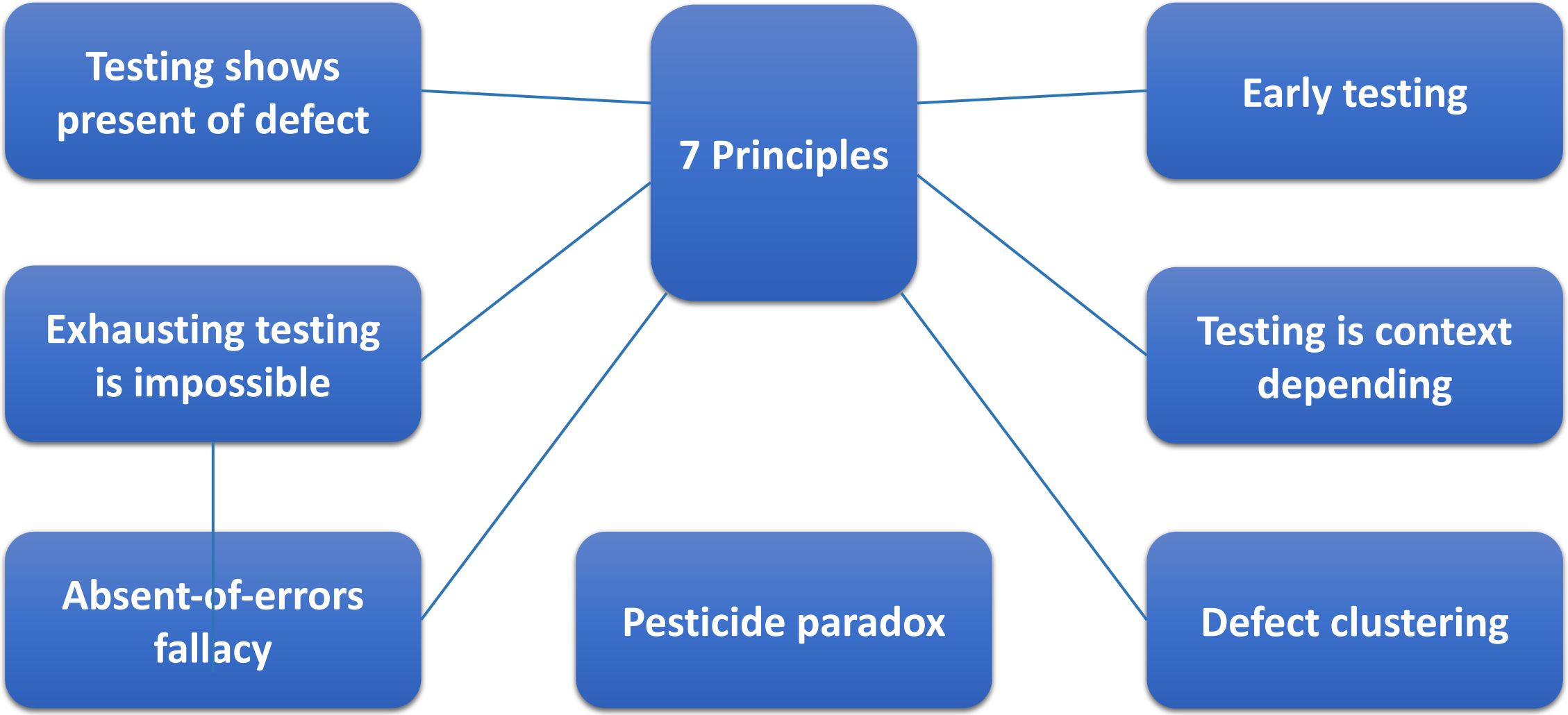
Тиждень 1. QA vs QC

	Quality Assurance	Quality Control
Визначення	Сукупність дій, спрямованих на те, щоб переконатись, що якість дотримується на всіх етапах розробки	Сукупність дій, спрямованих на те, щоб переконатись, що кінцевий продукт відповідає вимогам
Ціль	Ціллю є попередження дефектів шляхом покращення і дотримання процесів розробки	Ціллю є пошук (і виправлення) дефектів у вже існуючому продукті
Мета	Мінімізація / недопущення появлення дефектів на етапі розробки	Виявлення дефектів під час розробки і усунення їх до випуску продукту
Як?	Запровадження системи управління якістю (QMS), а також періодичні аудити її ефективності	Пошук та виявлення джерела помилок для їх усунення, для подальшої відповідності узгодженим вимогам

Тиждень 1. Тестування

Тестування –
викання тестових випадків і продукту, порівняння очікуваного і отриманого результату з метою виявлення дефектів

Тиждень 1. Принципи тестування

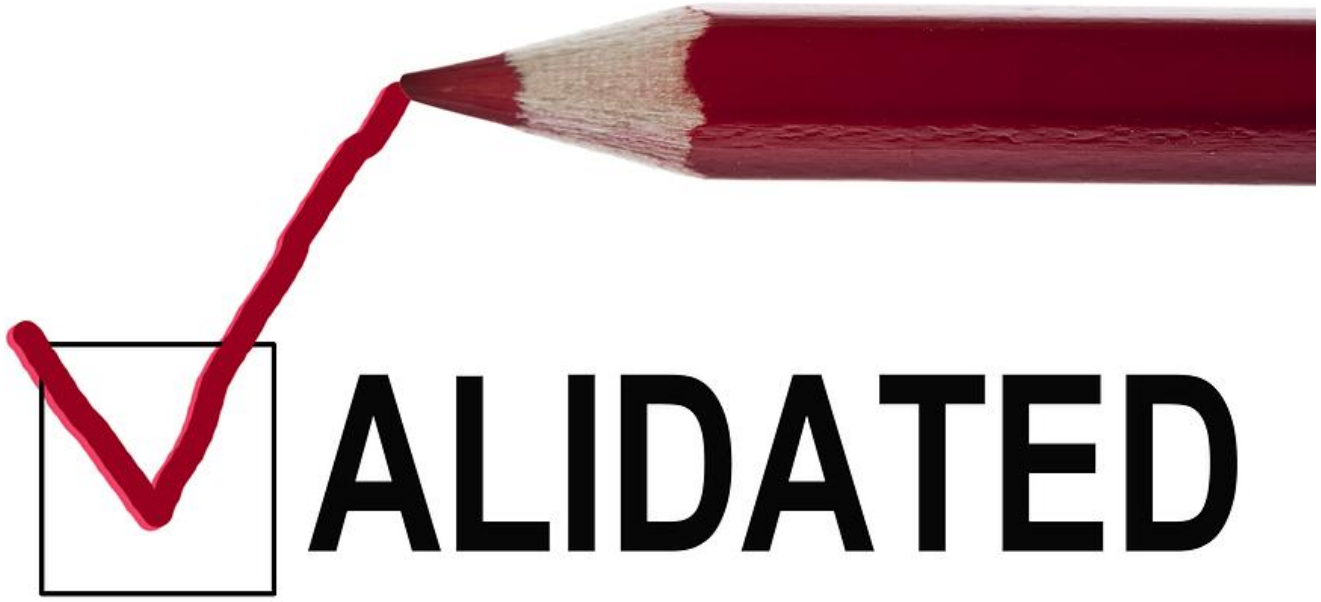


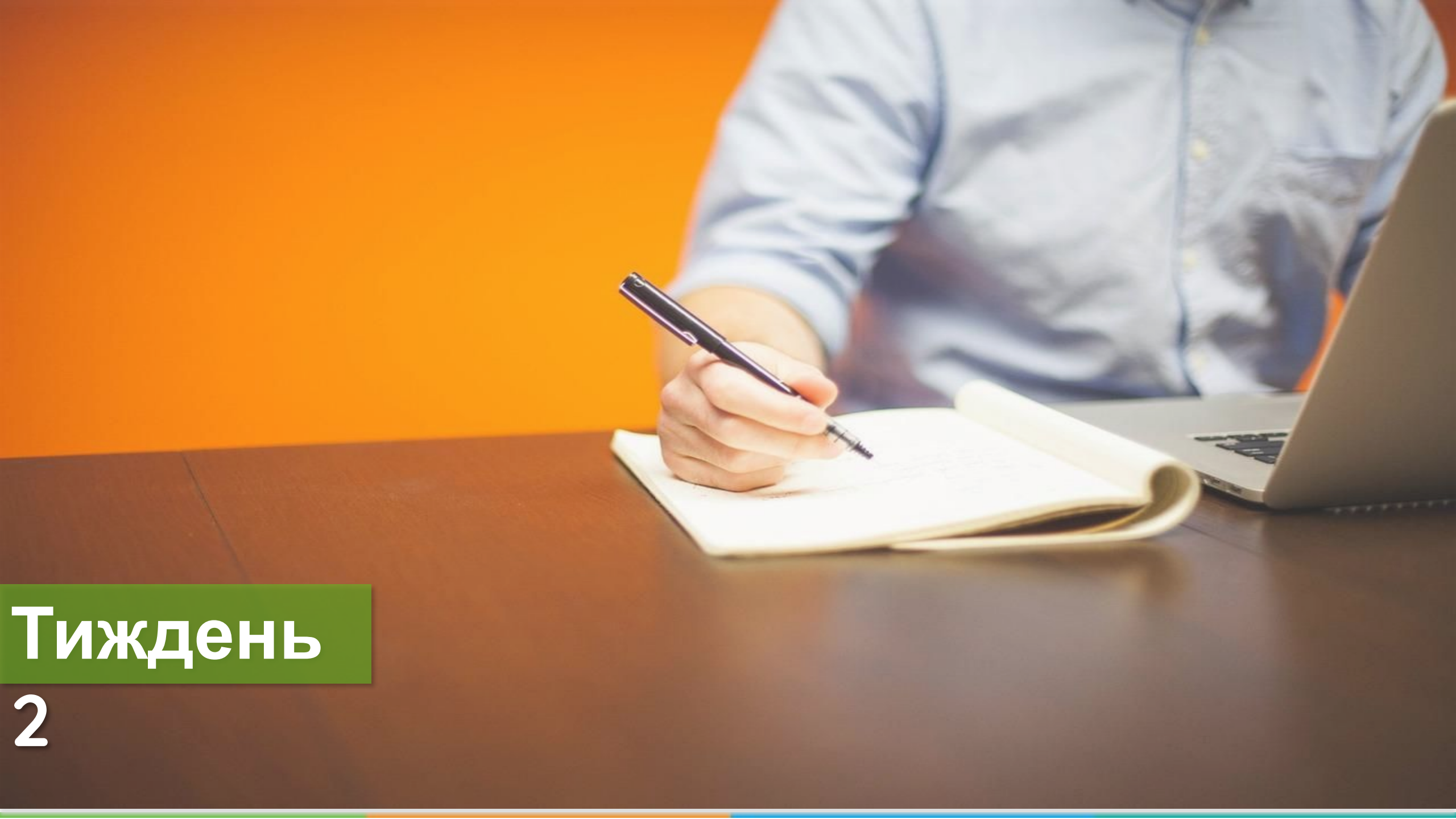
Тиждень 1. Верифікація

Верифікація – процес оцінки програмного забезпечення для визначення відповідності на певному етапі розробки, по відношенню до умов, накладених на початку цього етапу

Тиждень 1. Валідація

Валідація –
процес оцінки програмного забезпечення протягом процесу розробки, або зазвичай в кінці, для оцінки його відповідності визначеним вимогам

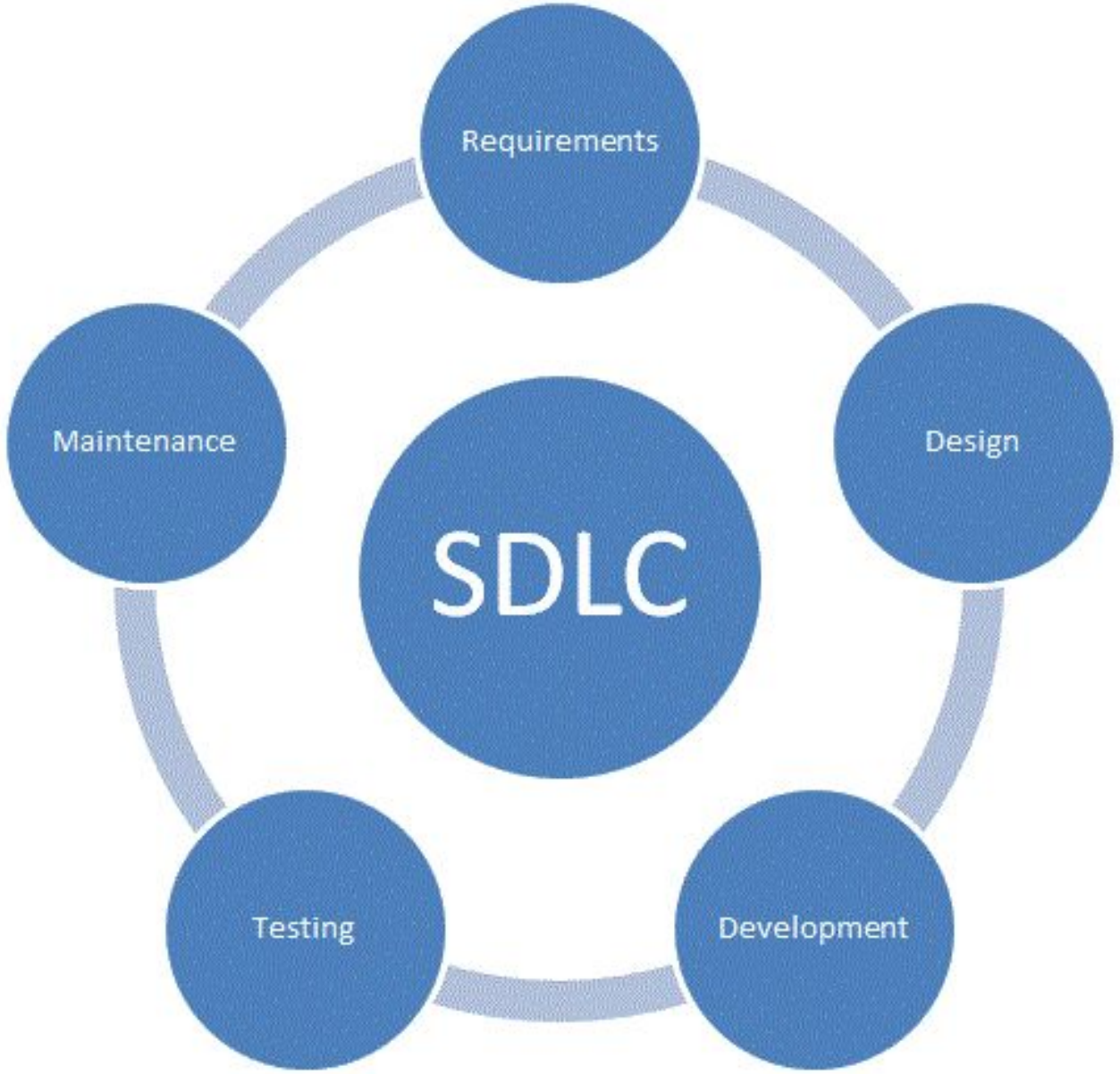




Тиждень 2

Тиждень 2. Життєвий цикл розробки ПЗ

1. Збір та аналіз вимог
2. Дизайн
3. Розробка
4. Тестування
5. Підтримка



Тиждень 2. Моделі розробки ПЗ

Інкрементальна модель

процес розробки, при якому вимоги розділяються на кілька частин, кожна з яких доставляється з наступним випуском

Тиждень 2. Моделі розробки ПЗ

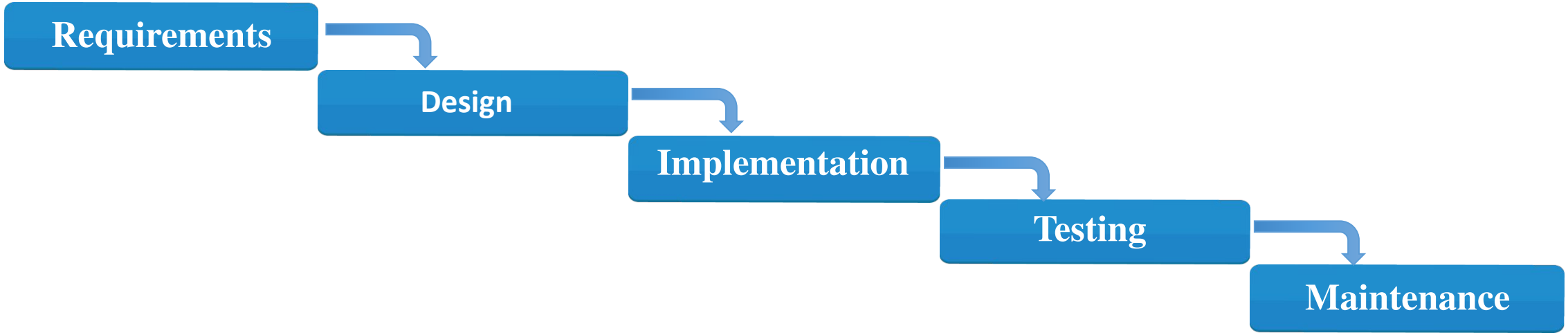
Ітеративна модель

процес розробки, при якому немає чіткого бачення кінцевого продукту, натомість подальші вимоги узгоджуються в процесі розробки

Тиждень 2. Waterfall

Waterfall

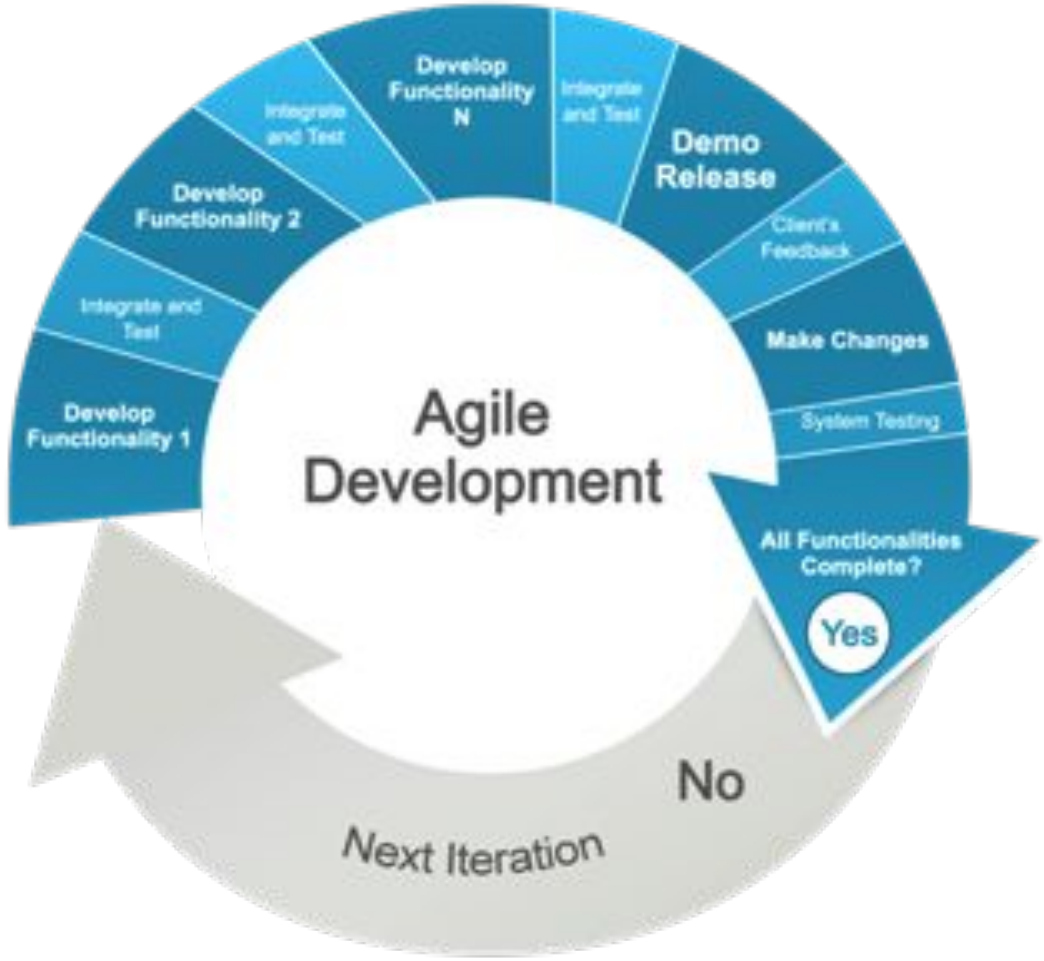
Методологія, при якій етапи розробки йдуть послідовно, кожна наступна починається по завершенні попередньої



Тиждень 2. Agile

Agile

Підхід в розробці програмного забезпечення, при якому використовується ітеративна модель, динамічне формування вимог, а також повна взаємодія в середині команди



Тиждень 2. Планування і контроль

Основні завдання

- Визначення обсягів і ризиків
- Визначення цілей тестування та підходів тестування
- Визначення необхідних ресурсів
- Планування тест дизайну, впровадження та виконання тестування
- Визначення ”вихідних” критеріїв
- Створення, узгодження та поширення тест плану

Артефакти

- Test Plan
- Configuration Matrix
- Test Hardware List
- Test Strategy

Тиждень 2. Аналіз та дизайн

Основні завдання:

- Оцінка основи для тестування
- Визначення тестових умов
- Дизайн тестових випадків
- Оцінка можливості тестування вимог та системи
- Розробка тестових даних, налаштування та конфігурація тестового обладнання

Артефакти:

- Test Design Specification
- Peer Review Records

Тиждень 2. Впровадження та виконання

Основні завдання впровадження:

- Розробка та пріоритизація тестових випадків
- Створення тестових комплектів для ефективного виконання
- Остаточне впровадження оточення

Основні завдання виконання:

- Виконання тест комплектів і окремих тестів
- Перевиконання попередньо неуспішних тестів
- Порівняння активний/очікуваний результат
- Звітування відмінностей як дефект
- Збереження результатів тест виконання

Артефакти:

- Test Cases
- Defect Reports

Тиждень 2. Оцінка результатів

Основні завдання:

- Перевірка чи результати задовільняють вихідні критерії
- Оцінка чи не потрібно додаткових тестових випадків
- Збір та аналіз щільності дефектів
- Написання фінального звіту для зацікавлених осіб

Артефакти:

- Звіт результату, що включає статус тестування, метрики, аналіз та заключення

Тиждень 2. Завершення тестування

Основні завдання:

- Перевірка, чи все що мало бути зроблено, виконано
- Переконатись, що всі дефект репорти проаналізовані
- Завершити та зберегти об’єкти що використовувались в тестуванні: тест скрипти, тестові дані та інше
- Передача тестового обладнання при здачі проекту
- Оцінка того як проходило тестування та засвоєння отриманих уроків для майбутніх проектів

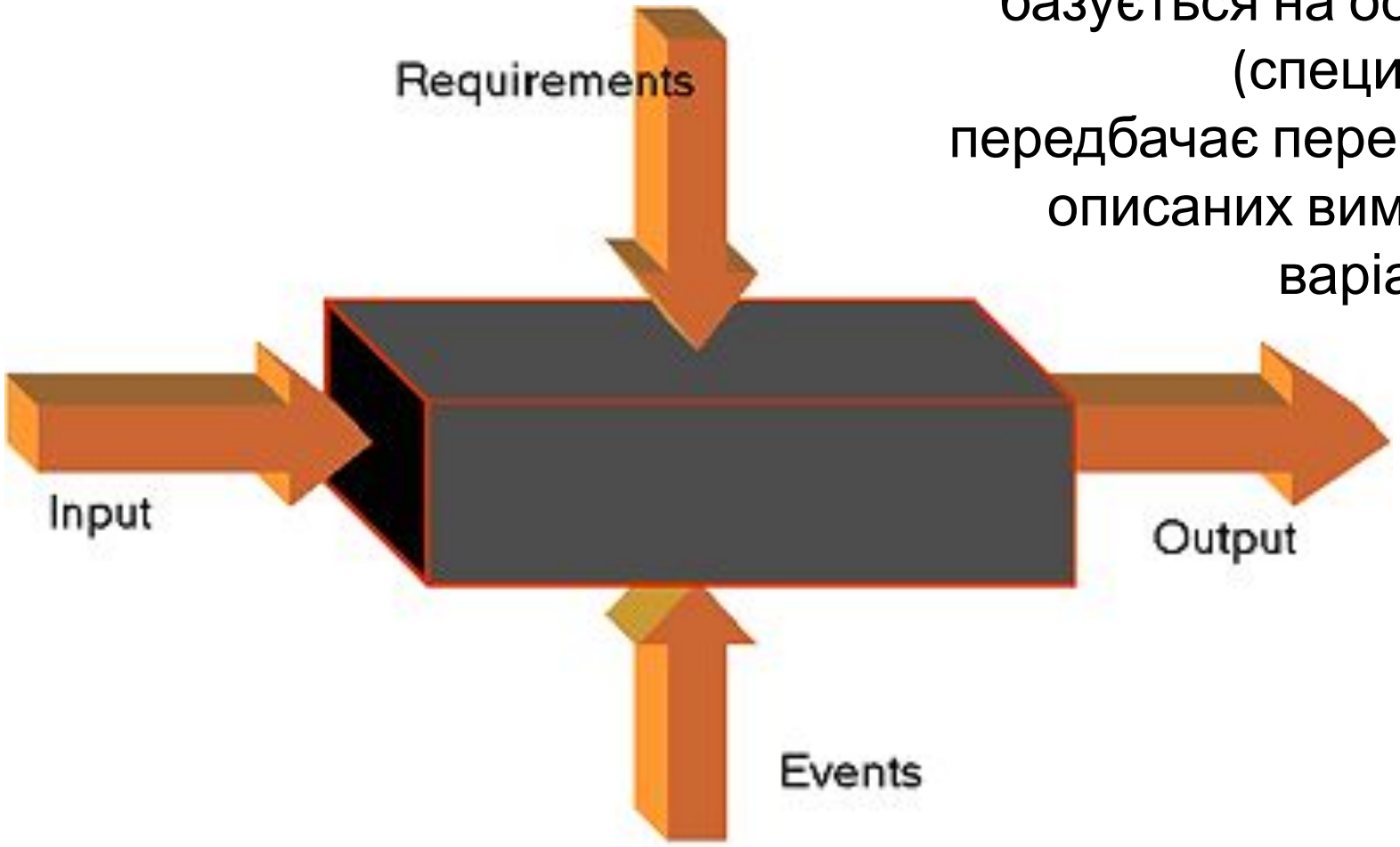
Тиждень 3

Тиждень 3. Типи тестування

Тиждень 3. Типи тестування

1. Functional testing (функціональне)
2. Non-functional testing (нефункціональне)
3. Structural testing (тестування структури/архітектури)
4. Testing related to changes (Re-Testing, Regression testing) (конфірмаційне, регресивне тестування)

Тиждень 3. Functional testing (функціональне тестування)



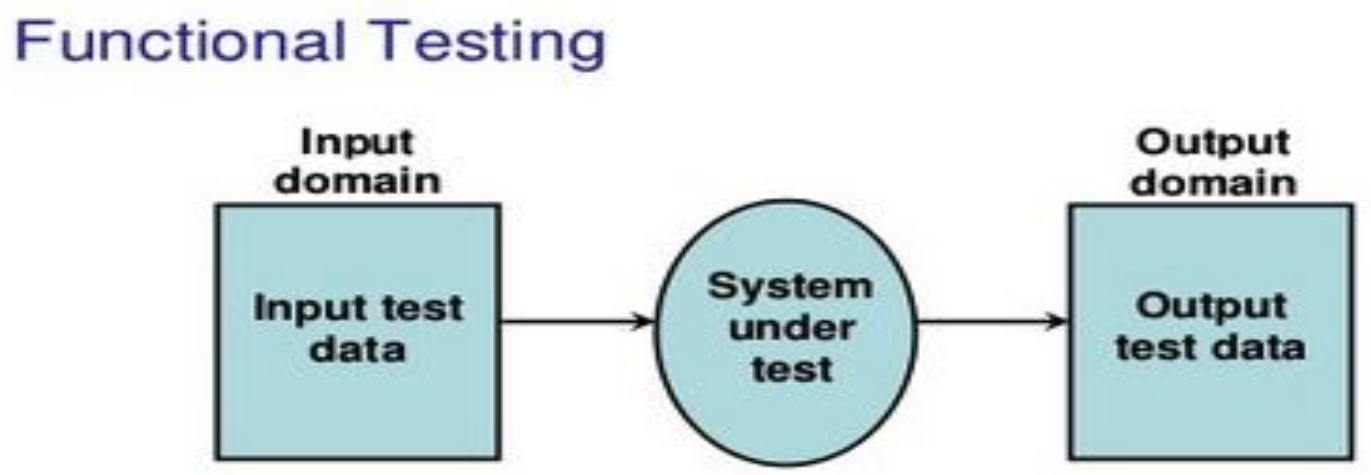
Функціональне тестування

базується на основі функціональних вимог (специфікації, інших видів вимог) і передбачає перевірку виконання програмою описаних вимог або розуміння можливих варіантів використання системи тестувальником.

Тиждень 3. Functional testing (функціональне тестування)

Основні задачі функціонального тестування:

1. Визначення ключових функцій / операцій системи, що знаходиться під тестуванням
2. Визначення змінних / вхідних даних, що використовуються системою при її роботі, та визначення границь цих змінних
3. Визначення змінних оточення / обладнання, що можуть вплинути на роботу системи, що знаходиться під тестування

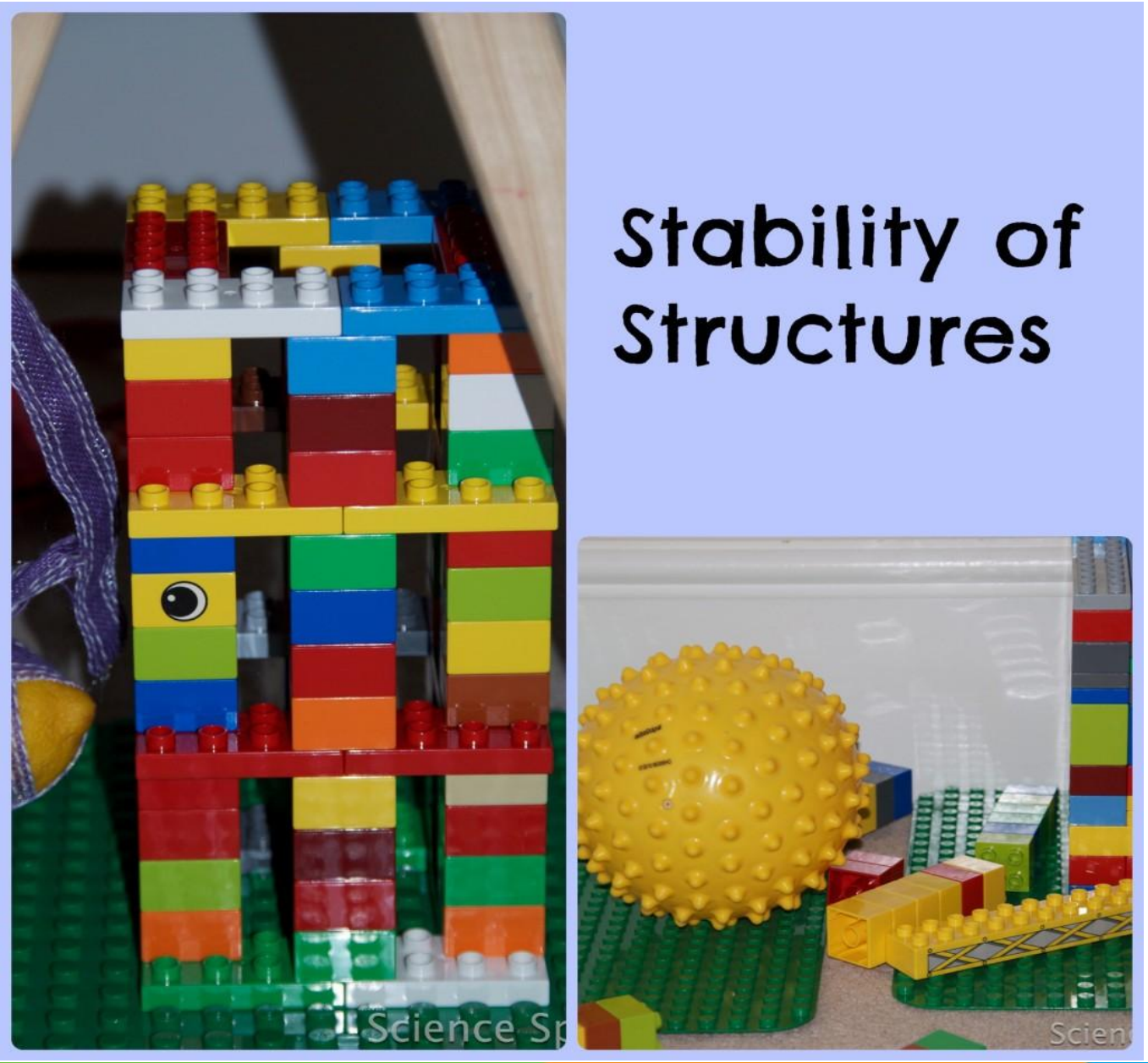


Тиждень 3. Non-functional testing (нефункціональне тестування)

Термін **нефункціональне тестування** описує тести (перевірки), які необхідні для вимірювання характеристик системи і програмного забезпечення, що можуть бути визначені кількісно по тій чи іншій шкалі, наприклад, час відгуку для тестування продуктивності.

- Performance testing
- Recovery testing
- Compatibility testing
- Localization testing

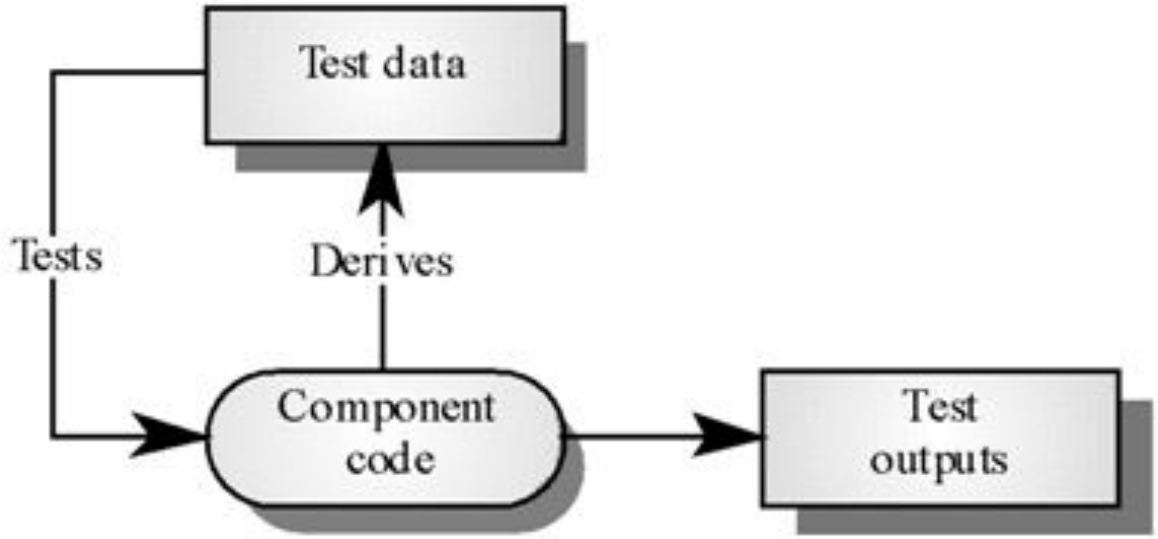
Тиждень 3. Structural testing (тестування структури/архітектури)



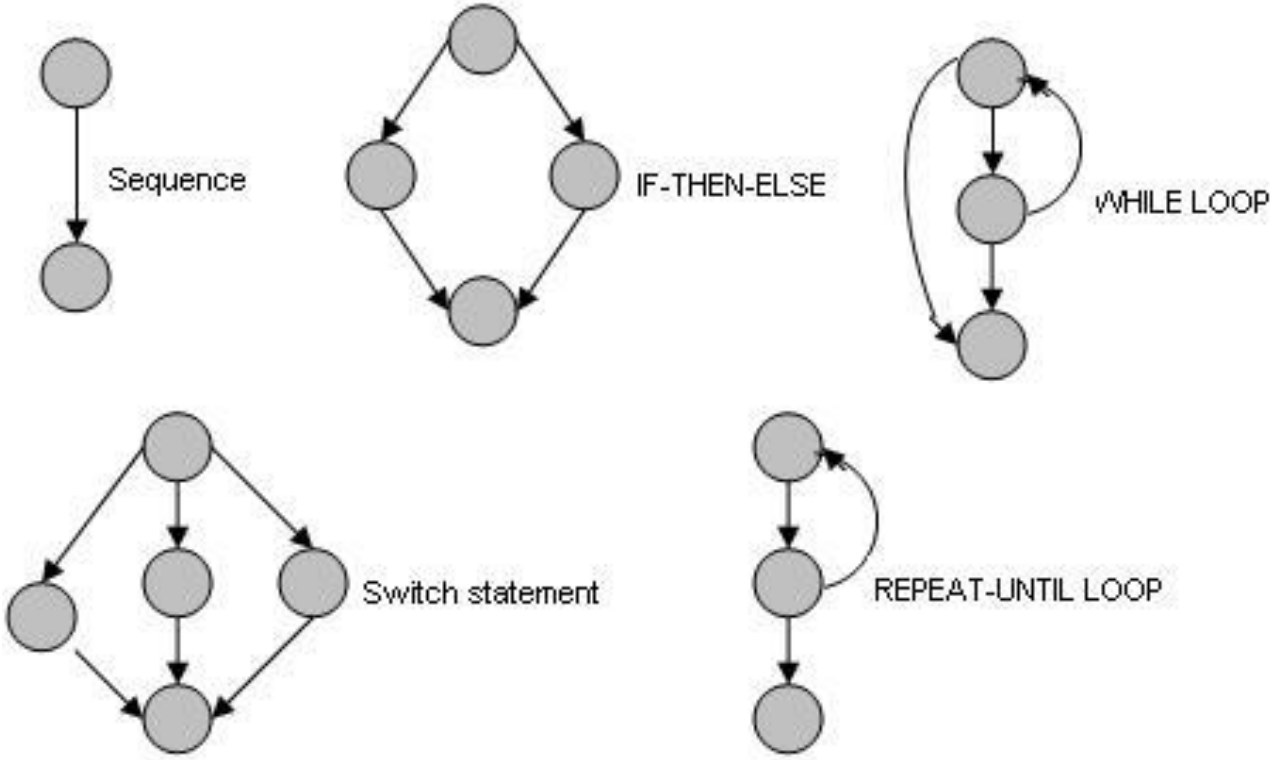
Структурне тестування, також відоме як тестування білого ящика (white-box(glass box)), є підхід, при якому тести походять від знання структури програмного забезпечення або внутрішньої реалізації.

Інші назви структурного тестування включає в себе clear box testing, open box testing, logic driven testing або path driven testing.

Тиждень 3. Structural testing (тестування структури/архітектури)



Тестування базується на основі знань про структуру програми



Структурні елементи, що описуються послідовностями

Regression Testing vs Retesting

Тиждень 3. Re-testing (confirmation testing)

Після того, як дефект був виявлений і виправлений, програмне забезпечення повинно бути протестовано ще раз , щоб підтвердити, що вихідний дефект був успішно виправлений.
Це називається **підтверджуючим тестуванням (re-testing / confirmation testing)**.

Тиждень 3. Regression testing (регресивне тестування)

Регресійне тестування є повторним тестуванням вже раніше протестованої програми, після будь яких модифікацій (зміни в коді, виправлення дефектів або зміна в оточуючому середовищі), щоб виявити будь-які дефекти, що можуть виникати внаслідок цих змін.

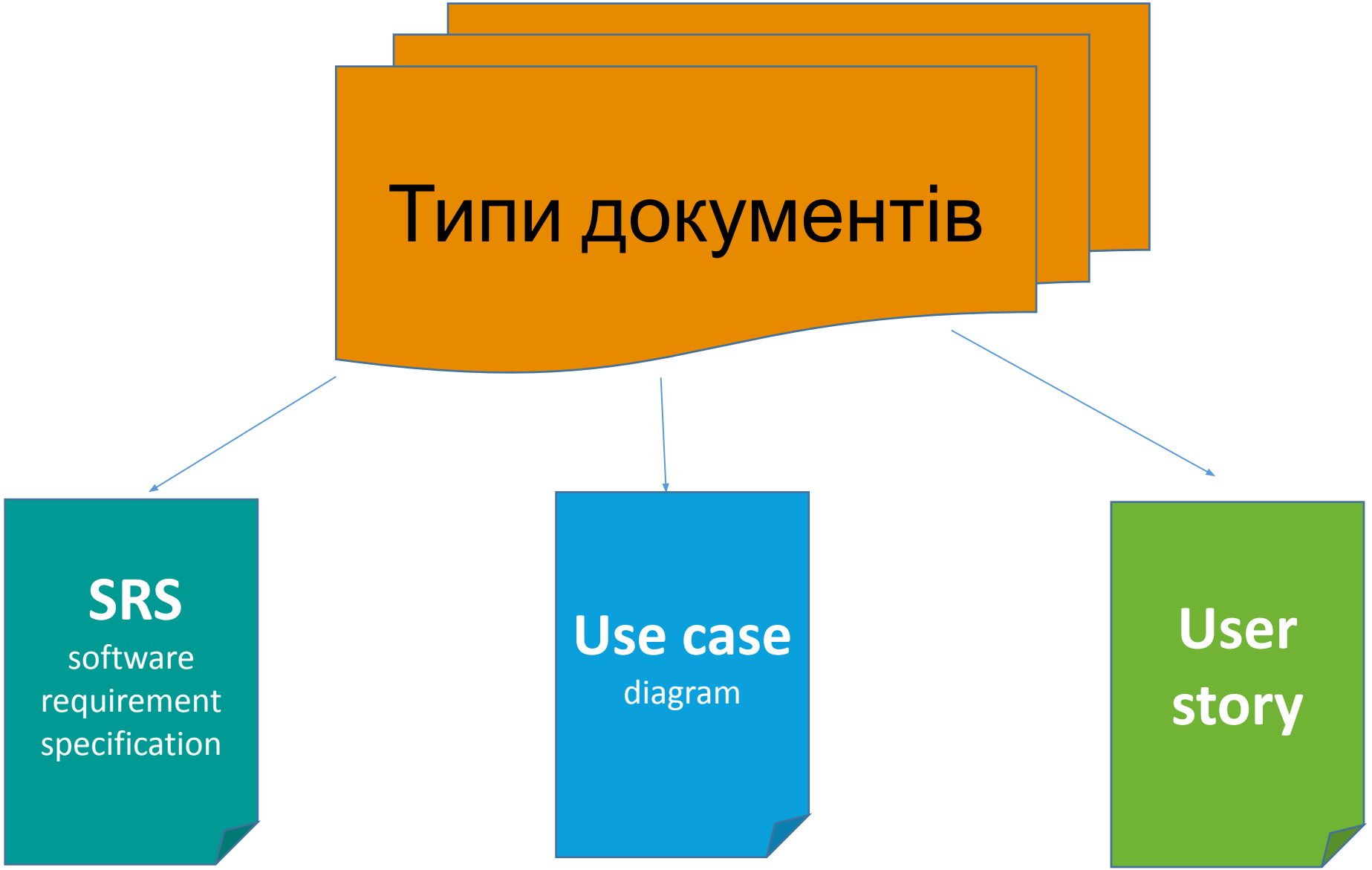
Regression Testing

99 little bugs in the code.
99 little bugs in the code.
Take one down, patch it around.

127 little bugs in the code...

Requirements Analysis

Тиждень 3. Аналіз вимог



Тиждень 3. SRS

Project Acronym	Business Requirements	Your Logo
Customer Name	Project ID No.:	

Table of Contents

1 Business Requirements 1
1.1 Purpose of the System 1
1.2 Problem Definition 1
1.3 Mandatory Requirements of the System 1
1.3.1 Mandatory Requirement #1 1
1.3.2 Mandatory Requirement #2 1
1.3.3 More Mandatory Requirements as needed 2
1.4 Optional Requirements of the System 2
1.4.1 Optional Requirement #1 2
1.4.2 Optional Requirement #2 2
1.4.3 More Optional Requirements as needed 2
1.5 Functional Requirements of the System 2
1.5.1 Functional Requirements in General 2
1.5.2 Functional Requirement #1 2
1.5.3 Functional Requirement #2 2
1.5.4 More Functional Requirements as needed 3
1.6 Constraints 3
1.7 Assumptions 3
1.8 Exclusions 3
1.9 Issues to be Resolved 3
2 Impact Statement 3
3 Risk Statement 3
4 Appendix 3

Software requirement specification (SRS) – описує, що повинно бути розроблено для системи (включаючи функціональні та нефункціональні вимоги)

Тиждень 3. Use case

письмовий описи взаємодії користувача з програмним продуктом для досягнення мети

Є способом різних варіантів дій, в яких може бути застосована система (функції, які система надає своїм користувачам)

Прецеденти допомагають нам виявити вимоги до документації

Тиждень 3. Use case складові

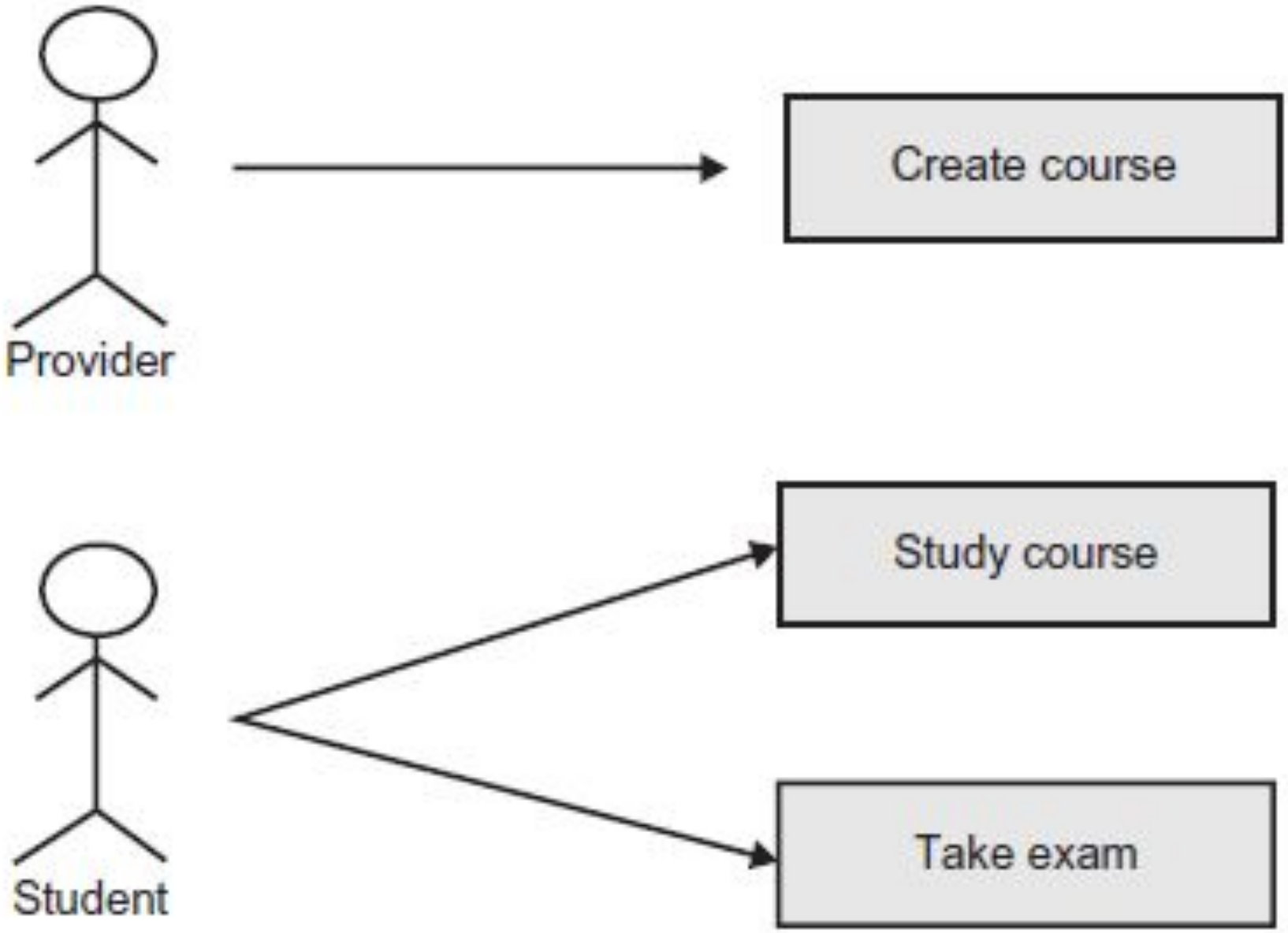
Use case має 3 компоненти:

- Завдання use case, що відображає функцію, яка повинна бути розроблена.

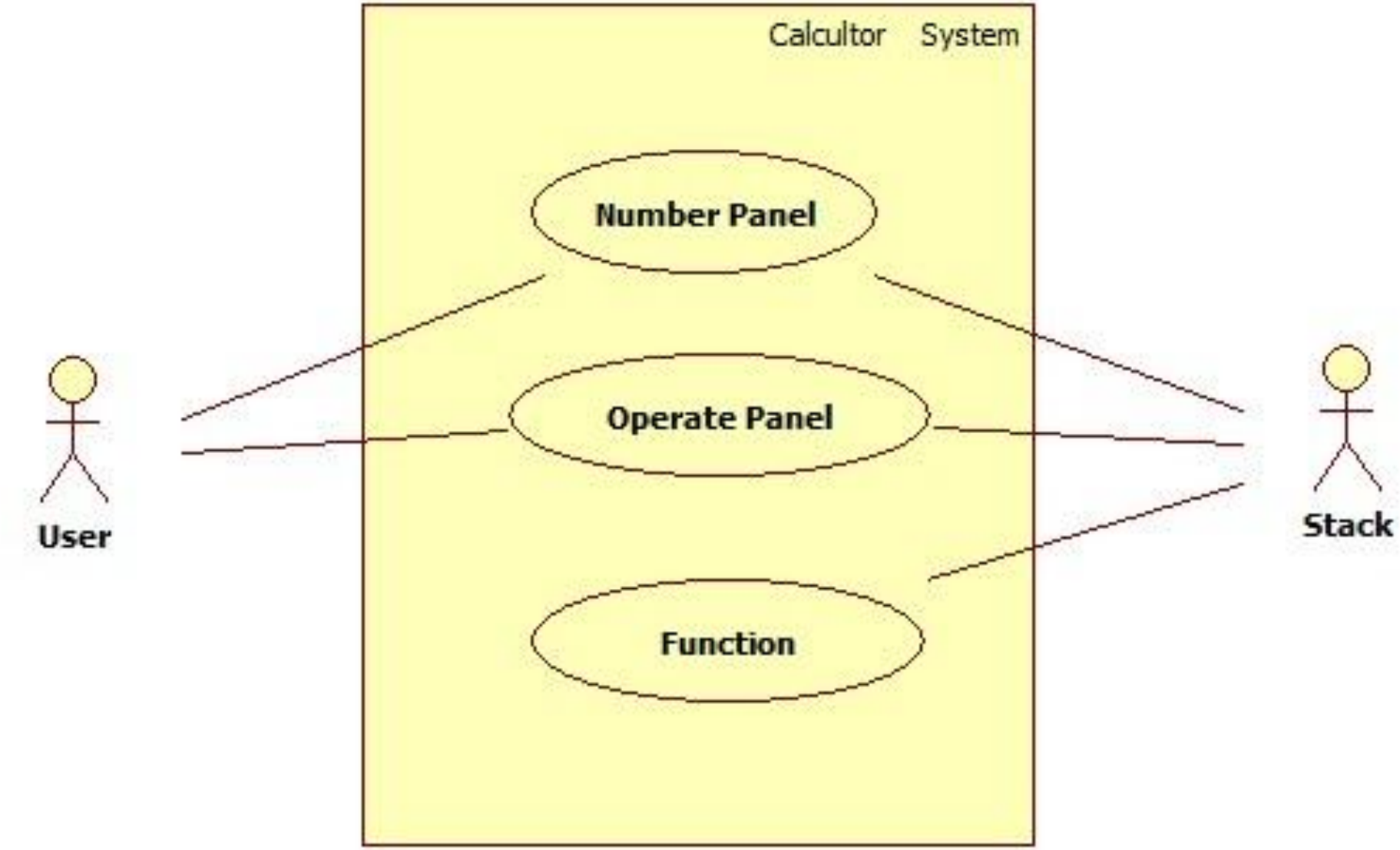
- Актори, що виконують цей use case.

- Лінія комунікації, що відображає яким чином актори комунікують з системою.

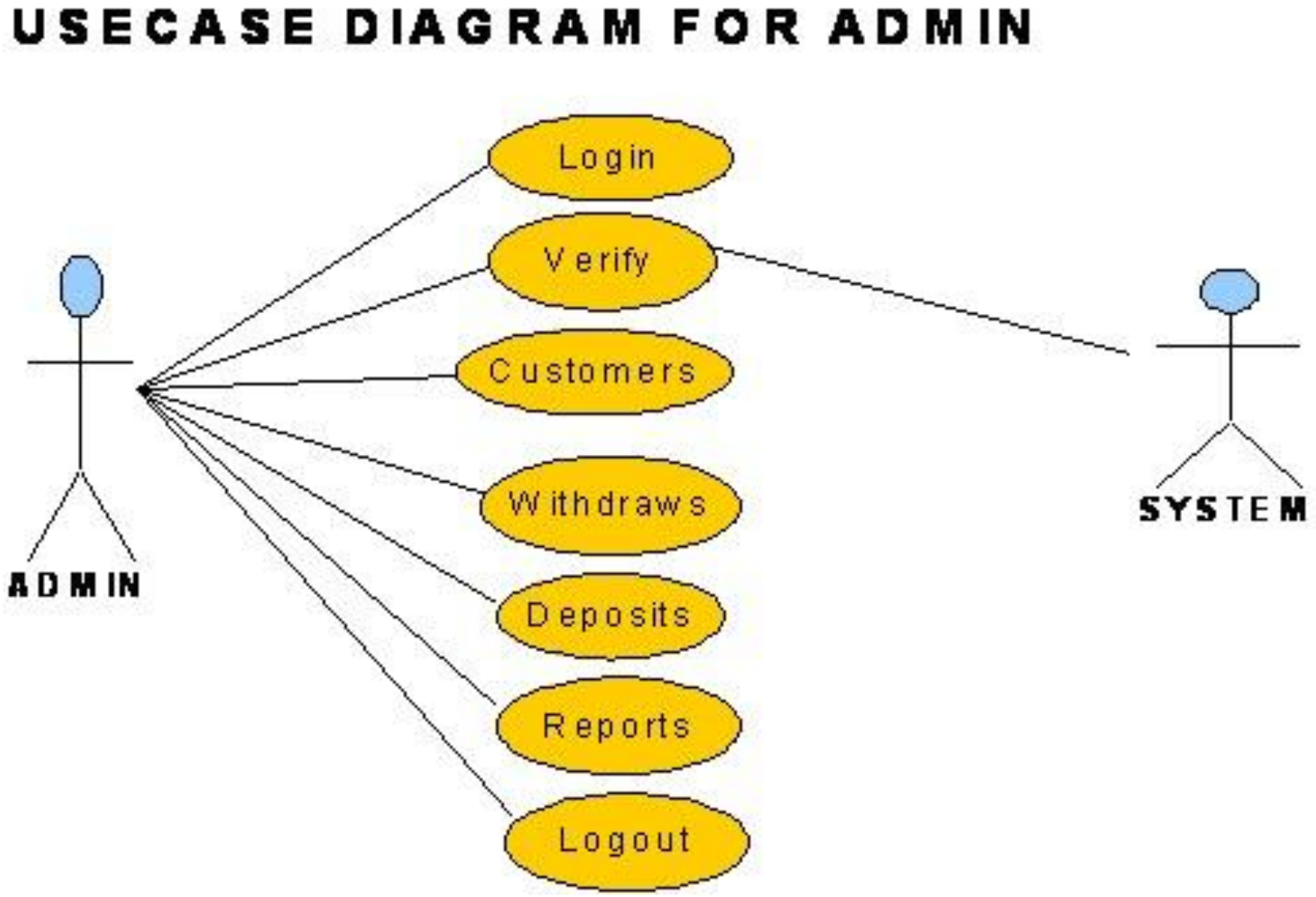
Тиждень 3. Use case. Приклад



Тиждень 3. Use case. Приклад



Тиждень 3. Use case. Приклад



Тиждень 3. Use case. Приклад

	Step	Description
Main Success Scenario **A: Actor** **S: System**	1	A: Inserts card
	2	S: Validates card and asks for PIN
	3	A: Enters PIN
	4	S: Validates PIN
	5	S: Allows access to account
Extensions	2a	Card not valid S: Display message and reject card
	4a	PIN not valid S: Display message and ask for re-try (twice)
	4b	PIN invalid 3 times S: Eat card and exit

Тиждень 3. Use case. Ключові елементи

Use Case Element	Description
Use Case Number	ID to represent your use case
Application	What system or application does this pertain to
Use Case Name	The name of your use case, keep it short and sweet
Use Case Description	Elaborate more on the name, in paragraph form.
Primary Actor	Who is the main actor that this use case represents
Precondition	What preconditions must be met before this use case can start
Trigger	What event triggers this use case
Basic Flow	The basic flow should be the events of the use case when everything is perfect; there are no errors, no exceptions. This is the "happy day scenario". The exceptions will be handled in the "Alternate Flows" section.
Alternate Flows	The most significant alternatives and exceptions

Тиждень 3. User story

Опис - письмовий опис користувацької історії як для цілей планування так і нагадування

Діалог - розділ для збору додаткової інформації про користувацьку історію та деталі будь-яких розмов

Підтвердження - розділ, щоб передати те, що тести будуть проведені, щоб підтвердити історію користувача, що вона є повною і працює, як очікувалося

Тиждень 3. User story

As...	Conditional	I want...	So that...
As an HR Rep	who is authorized to initiate reviews for new employees	I want to be notified when new hires have reached their 90 day mark	so that I can initiate a 90-day review.
As an HR Rep	who has initiated a 90-day review	I want to notify the new hire of all of the requirements of the 90-day review	so they can begin to submit their evaluation in the system.
As an HR Rep	who has initiated a 90-day review	I want to notify the new hire of all of the requirements of the 90-day review	so they can begin to submit their evaluation in the system.
As an employee	who is under 90-day review	I want to create a log in for the HR review system	so that I can log in to submit my 90-day evaluation.
As an employee	who is under 90-day review	I want to log in to the system	so that I can view the requirements for my 90-day evaluation.
As an employee	who is under 90-day review	I want to submit the names of two peers I have worked with since being hired	so that they can contribute to my 90-day review.
As an employee	selected to peer review a new hire	I want to be notified when I have been selected to review a new hire after 90 days	so that I can log in to the system and submit my evaluation.

Тиждень 3. Функціональні вимоги

Функціональні вимоги описують, які функції системи повинна виконувати.

Функціональні вимоги визначають конкретні дії, які необхідні для виконання цілей використання.

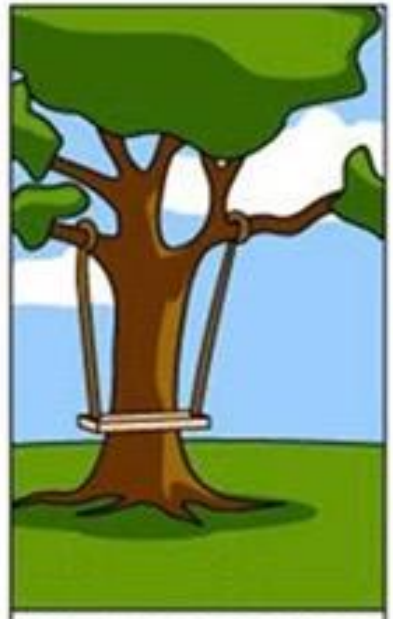
Тиждень 3. Нефункціональні вимоги

Нефункціональні вимоги визначають загальні властивості або атрибути отриманої системи.

Нефункціональні вимоги є обов'язковою вимогою програмного забезпечення, які описують не те, що програмне забезпечення буде робити, але, як програмне забезпечення буде робити це.

AS PROPOSED BY PROJECT SPONSOR

AS SPECIFIED IN PROJECT REQUEST

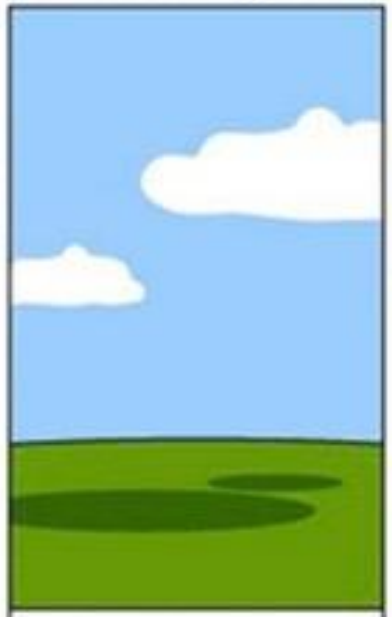


HOW PROJECT LEADER UNDERSTOOD IT

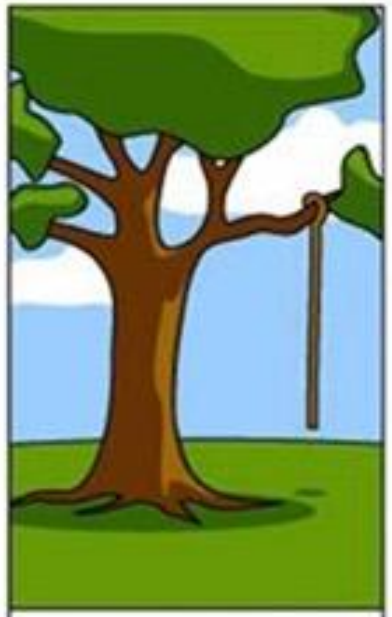
AS DESIGNED BY SENIOR ANALYST

AS PRODUCED BY THE PROGRAMERS

HOW CONSULTANT DESCRIPED IT

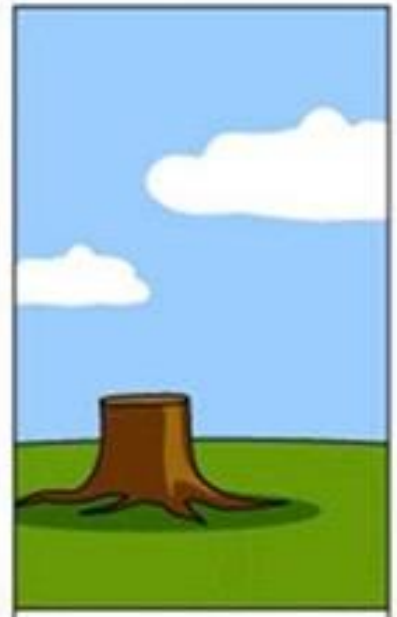


HOW PARTS WERE DOCUMENTED



WHAT WAS INSTALLED

HOW PROJECT WAS BUDGETED & BILLED



HOW USERS THINK IT'S SUPPORTED

WHAT THE CUSTOMER REALLY WANTED

Тиждень 3. Чекліст для вимог

- Чіткі та зрозумілі?
- Чи присутня двозначність?
- Відсутність невизначених займенників(e.g., this, these, it)?
- Висловлювання логічні та послідовні?
- Висловлює позитивні дії, замість негативних (e.g. ‘shell not’)?
- Чи може вимога бути неправильно витлумачена?
- Відсутність термінів, що неможливо перевірити, протестувати?
- Чи є технічна можливість реалізації даної вимоги?
- …

Тиждень 4

Тиждень 4. Test case anatomy.
Title

Title (Header) – повна назва тест кейсу яка відображає зміст перевірки, що буде виконана. Досить часто використовується в якості елемента чекліста, саме тому варто максимально чітко та стисло описувати.

Неправильно:
Verify that user can’t be logged in with incorrect credentials

Правильно:
Verify that user can’t be logged in with incorrect username and correct password
Verify that user can’t be logged in with correct username and incorrect password
…..

Тиждень 4. Test case anatomy

- Title (Header)
- ID (test case №)
- Description
- Steps
- Expected results
- Status (passed, failed, not executed, blocked)
- Priority (Smoke, Critical, Extended; or A, B, C, D or any other)
- Initial data we need for test
- Related requirement
- Module, submodule
- Author, last time run, actual result, related bug
- Estimate TTR

Тиждень 4. Test case anatomy. Description

Поле **Description** заповнюється з метою надання максимальної інформації стосовно даної перевірки. Будь які передумови або інші умови за яких повинна бути здійснена перевірка описується тут.

Тиждень 4. Test case anatomy. Steps

Поле “Кроки” може бути як незалежним елементом, так і частиною блоку Description.
Тут описуються кроки перевірок, що будуть відбуватись.

Приклад:

1. Go to Home page as logged in user.
2. Click "Sitemap" link.
3. Click Privacy Policy, Terms & Conditions and Cookie Policy hyperlinks one-by-one.
4. Repeat #1-3 steps for not logged user account.

Кроки повинні бути максимально зрозуміло описані, аби людина яка буде залучена до тестування в майбутньому змогла легко відтворити дані кроки. Варто звернути увагу, що деталізація кроків не повинна починатись з кроків ”1. Підійдіть до комп’ютера. 2. Включіть комп’ютер і т. д.”

Тиждень 4. Test case anatomy. Expected result

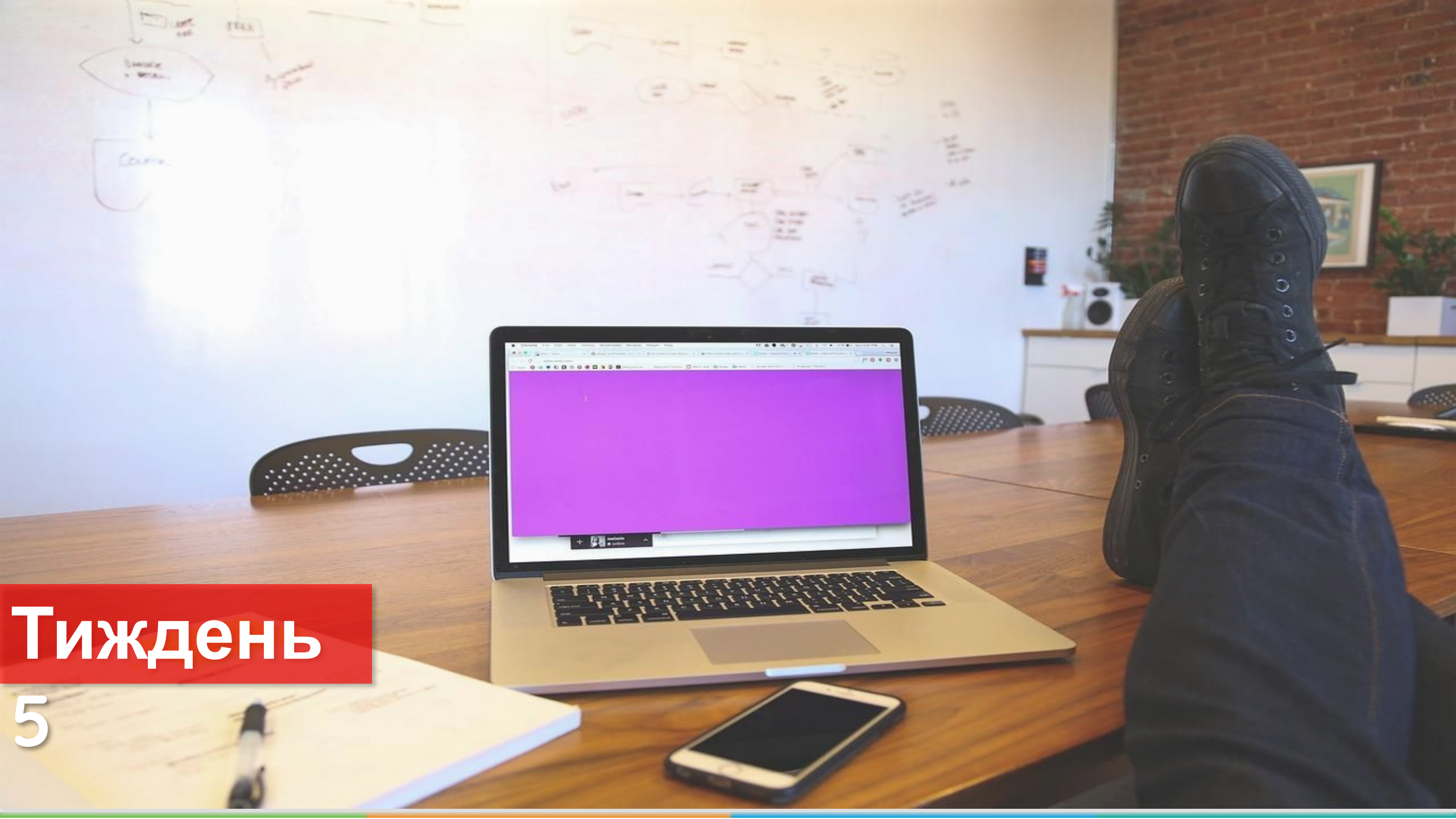
В блоці очікуваного результату описується еталонний результат поведінки системи на певні дії, з яким в подальшому буде порівнюватись очікуваний результат:

Правильно:
User should see Contact the center page with Request call back and Message to center forms.

Неправильно:
User should see Contact the center page with elements.

Тиждень 4. Test case anatomy. Other fields

Status (passed, failed, not executed, blocked) – статус виконання тест кейсу
Priority (P1,P2,P3,P4 or A, B, C, D or any other) – визначає пріоритет виконання тест кейсу
Initial data we need for test – дані, необхідні для виконання тест кейсу (зареєстрований користувач при перевірці функціональності логування в систему)
Related requirement – для можливості відслідковування покриття вимог тестами
Module, submodule – модуль / підмодуль, якого стосується даний тест
Author – людина, що створила тест кейс
last time run – час останнього запуску тест кейсу
related bug – для можливості відслідковування дефектів пов’язаних з даним тестом
Estimate TTR – оцінка затрат часу на виконання даного тесту



Тиждень 5

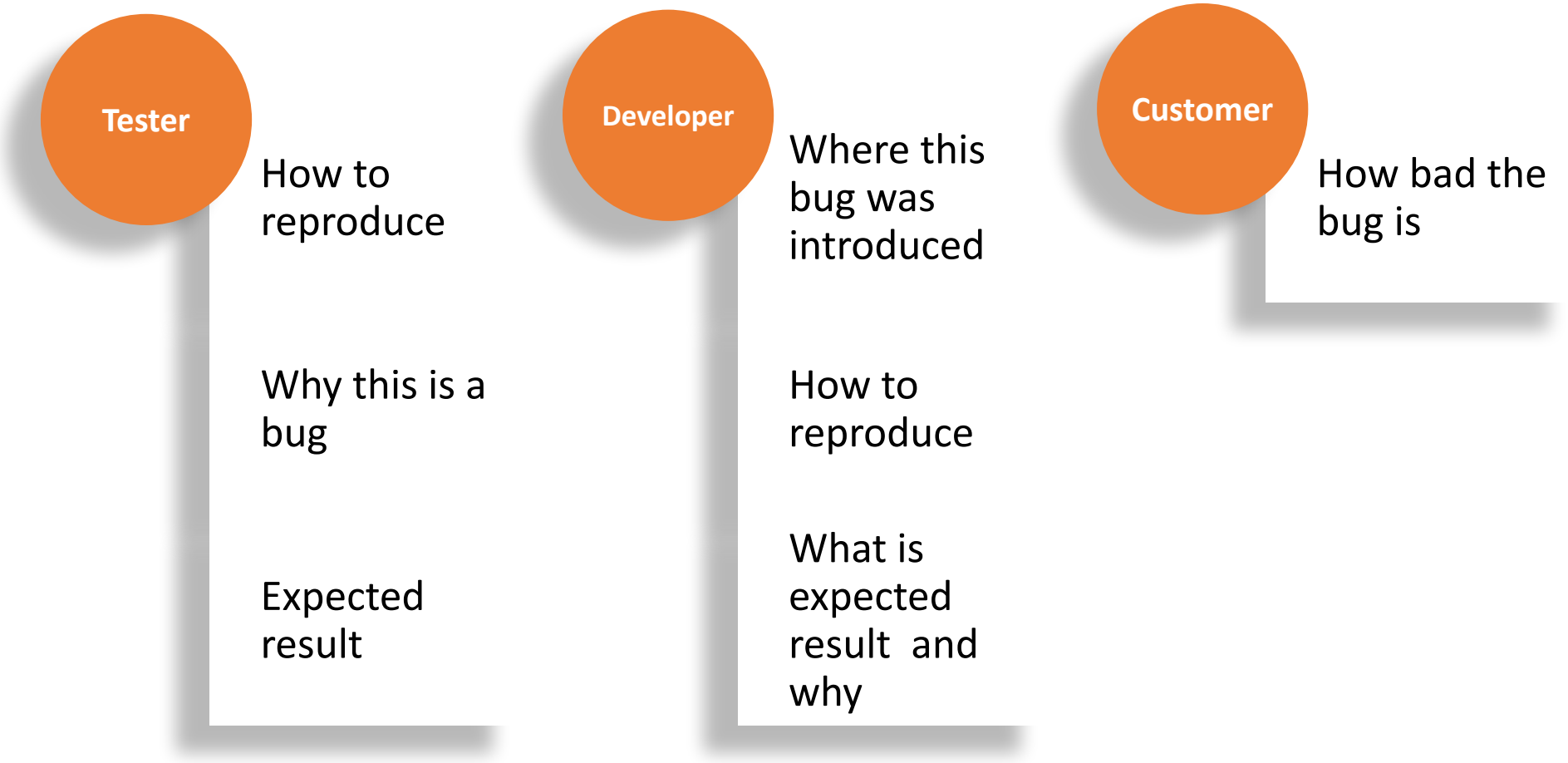
Тиждень 5. Найпоширеніші типи помилок

- Арифметичні помилки
- Логічні помилки
- Пов’язані з часом/датою
- Синтаксичні помилки
- Помилки пов’язані з ресурсами
- Multi-threading дефекти
- Дефекти пов’язані з інтерфейсами
- Performance дефекти

Тиждень 5. Хто може рапортувати дефект

- Тестувальники або QA персонал
- Розробники
- Працівники служби технічної підтримки
- Працівники відділу маркетингу по продаж
- Клієнти
- Кінцеві користувачі

Important in Defect Report for



Тиждень 5. Структура дефект репорту

Тиждень 5. Priority & Severity

Priority – визначає рівень впливу знайденого дефекту на бізнес, а відповідно вказує, наскільки критичним він є з точки зору кінцевого користувача і як швидко повинен бути виправлений

Можливі значення:

High – виправити якомога швидше

Medium – важливий, але менш важливий ніж поточне завдання

Low – виправлення може не відбуватись, або виправляється в останню чергу

Severity – визначає рівень впливу знайденого дефекту на систему. Іншими словами, наскільки критичний вплив даної помилки на інші частини системи або системи в цілому

Можливі значення:

Critical – повна втрата працездатності системи

Major – втрата даних або відмова у роботі певної частини функціоналу

Minor – некритичний функціонал не працює, відмова деяких функцій

Cosmetic – графічні дефекти, які не впливають на працездатність системи

Тиждень 5. Як правильно визначити пріоритет

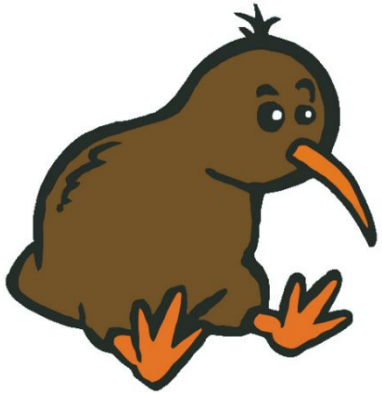
	100 % - 75 %	75 % - 50 %	50 % - 25 %	< 25 %
Crash	Priority 1	Priority 1	Priority 1	Priority 1
Non-Functioning	Priority 1	Priority 1	Priority 1	Priority 2
Incorrectly Functioning	Priority 1	Priority 1	Priority 2	Priority 2
Incorrectly Functioning with workaround	Priority 1	Priority 2	Priority 2	Priority 3 or 4
Performance	Priority 2	Priority 2	Priority 3 or 4	Priority 3 or 4
Cosmetic	Priority 2	Priority 3 or 4	Priority 3 or 4	Priority 3 or 4

Тиждень 5. Defect Tracking
Tools

JIRA

Target Process

Trac

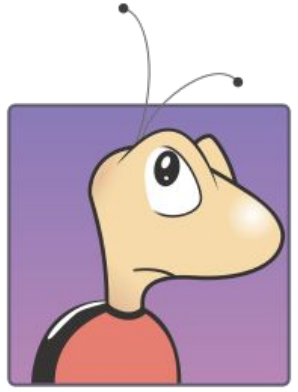


FogBugz

Team Foundation Server

TestTrack

Rally



Bugzilla