СУЧАСНІ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

122 «Комп’ютерні науки»
КН-18
2019 / 2020 навчальний рік

Технології # 5.
Якість & Тестування ПЗ

1. Дефекти ПЗ, визначення, їх класифікація.
2. Якість ПЗ.
3. Тестування, принципи, життєвий цикл дефекту.
4. Рівні, типи та види тестування.
5. Документування.

https://github.com/eabshkvprof/2020_Mod_Prog_Techn

Проблеми створення ПЗ

ПЗ веде себе невірно (що таке «веде» , що таке «невірно»?).

ЧОМУ? , ЯК ВИПРАВИТИ?

Помилка (*error*) - дія людини, яка породжує неправильний результат.

Дефект, баг (defect, bug) - недолік модуля (компонента) або системи, який може привести до відмови певної функціональності. Дефект, виявлений під час виконання програми, може викликати відмову окремого компонента або всієї системи.

Проблеми створення ПЗ

Результат дефекту: програма може не робити того, що повинна або навпаки - робити те, чого не повинна: відбувається →

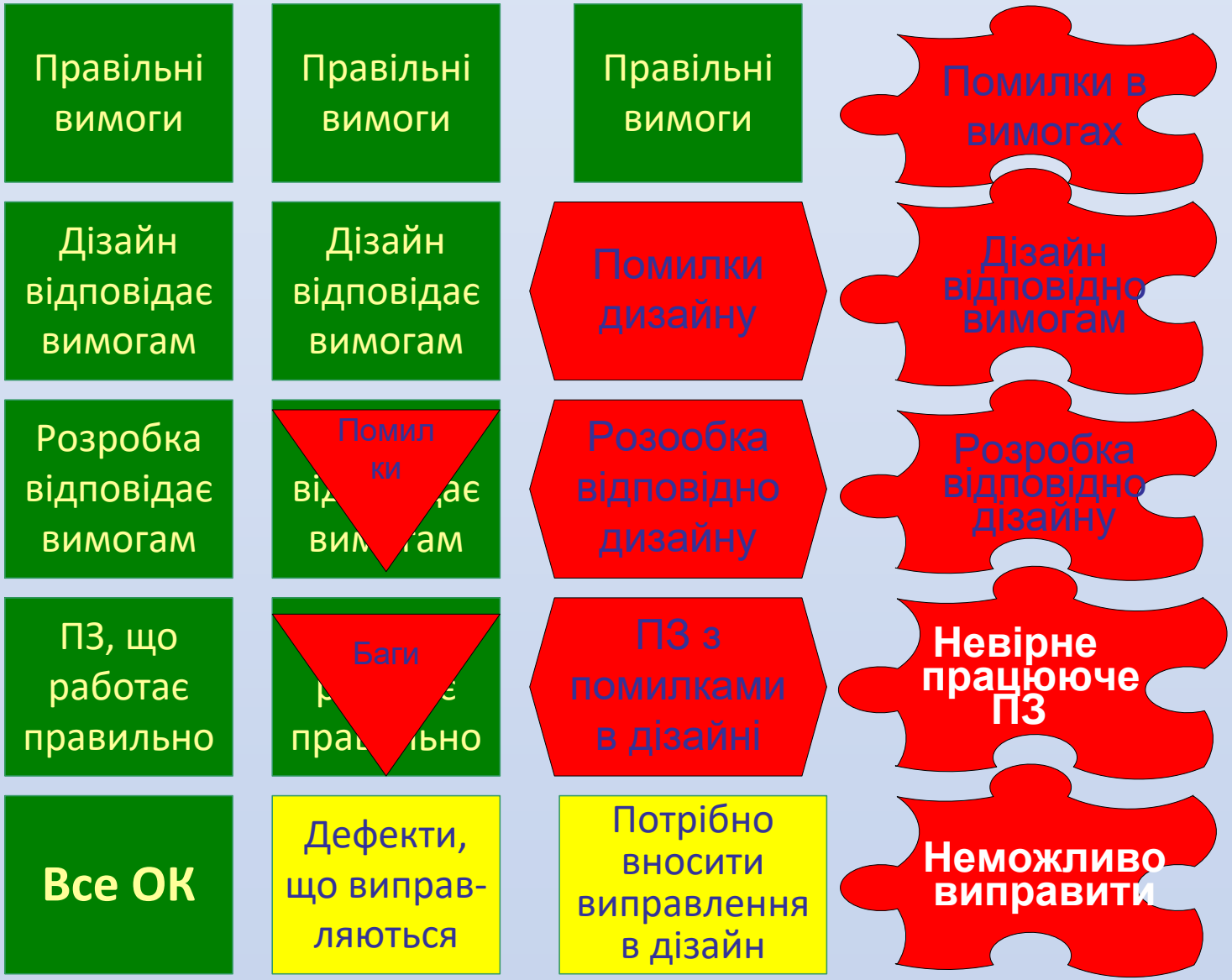
Збій (*failure*) - невідповідність фактичного результату (*actualresult*) роботи компонента або системи очікуваного результату (*expectedresult*).

Баг існує при одночасному виконанні трьох умов:

- **відомий очікуваний результат;**
- **відомий фактичний результат;**
- **фактичний результат відрізняється від очікуваного результату.**

Не всі баги є причина збоїв - деякі можуть ніяк себе не проявляти і залишатися непоміченими.
Причиною збоїв можуть бути не тільки дефекти, але також і умови навколишнього середовища.

Проблеми створення ПЗ



Причини дефектів

Недолік або відсутність спілкування.

! Замовник розуміє, яким він хоче бачити готовий продукт, але належним чином не пояснив його ідею розробникам і тестувальникам. Вимоги повинні бути доступні і зрозумілі всім учасникам процесу розробки ПЗ.

Складність програмного забезпечення.

Продукт складається з безлічі компонентів, які об'єднуються в складні програмні системи. Все складніше написання і підтримка. Чим складніше // важче робота, тим більше помилок може допустити виконуючий її чоловік.

Причини дефектів

Зміни вимог. Незначні зміни вимог на пізніх етапах розробки вимагають великого обсягу робіт по внесенню змін до системи. !!! Часта *зміна* вимог в бізнесі - *правило*, ніж виняток, тому безперервне тестування і контроль ризиків в таких умовах - прямий обов'язок розробника (відділ забезпечення якості).

Погано документований код. Складно підтримувати і змінювати погано написаний і слабо документований програмний код.

Засоби розробки ПО. Засоби візуалізації, бібліотеки, компілятори, генератори скриптів та інші допоміжні інструменти розробки - це теж часто погано працюють і слабо документовані програми, які можуть стати джерелом дефектів.

Необхідність тестування

Чим пізніше дефект буде виявлений, тим дорожче обійдеться його виправлення, тим більше зусиль для цього буде потрібно.

Висновок : чим раніше в життєвому циклі програми почнеться тестування, тим більшою мірою можна бути впевненим в якості ПЗ.

Якість ПЗ

Якість – «відповідність вимогам» & «придатність до використання».

Комплекс стандартів ISO 9000 – загальні принципи керування якістю продукції .

Стандарт ISO 90003:2004 – «Software engineering — Guidelines for the application of ISO 9001:2000 to computer software». Якість ПЗ - сукупність характеристик ПЗ, що відносяться до його здатності задовольняти встановлені і передбачувані потреби.

Якість ПЗ за ISO/IEC 25000:2014 – здатність ПЗ при заданих умовах задовольняти встановленим або передбачуваним потребам.

Якість ПЗ

Функціональність (*functionality*) **- визначає здатність ПЗ вирішувати завдання, які відповідають зафіксованим і очікуваним потребам користувача, при заданих умовах використання. Характеристика відповідає за те, що ПЗ працює справно і точно, функціонально сумісно, відповідає стандартам галузі і захищене від несанкціонованого доступу.**

Надійність (*reliability*) **- здатність ПЗ виконувати необхідні завдання в позначених умовах протягом заданого проміжку часу або вказану кількість операцій. Атрибути - завершеність і цілісність всієї системи, здатність самостійно і коректно відновлюватися після збоїв в роботі, відмовостійкість.**

Якість ПЗ

Зручність використання (*usability*) / супроводу (*maintainability*) - можливість легкого розуміння, вивчення, використання і привабливості ПЗ для користувача. Легкість, з якою ПО може аналізуватися, тестуватися, змінюватися для виправлення дефектів,.

Ефективність (*efficiency*) - здатність ПЗ забезпечувати необхідний рівень продуктивності у відповідність з виділеними ресурсами, часом і іншими позначеними умовами.

Мобільність (портативність, *portability*) - характеризує ПО з точки зору легкості його перенесення з одного оточення (*software / hardware*) в інше.

Модель якості ПЗ (ISO 9126-1)

Якість ПЗ

- **1. Функціональність**
  - Відповідність стандартам
  - Функціональна сумісність
  - Функціональна справність
  - Безпечність
  - Точність
- **2. Надійність**
  - Стійкість до відмов
  - Завершенність
  - Відновленність
- **3. Ефективність**
  - **Часова ефективність**
  - **Ефективність використання ресурсів**
- **4. Зручність використання. Простота**
  - Встановлення
  - Вивченння
  - Зрозумілість
- **5. Зручність супроводу**
  - Аналізуємость
  - Стабільність
  - Змінність

Тестування ПЗ

Тестування ПЗ це

- процес дослідження ПО з метою отримання інформації про якість програмного продукту;
- процес перевірки відповідності заявлених до продукту вимог і реально реалізованої функціональності, здійснюваний шляхом спостереження за його роботою в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином;
- оцінка системи з тим, щоб знайти відмінності між тим, який система повинна бути і якою вона є.

Тестування - це одна з технік контролю якості (Quality Control), яка включає **планування**, **складання** тестів, безпосередньо **виконання** тестування і **аналіз** отриманих результатів.

Тестування ПЗ

!!!! Тестування ПЗ включає не тільки власне проведення тестів, але і дії, пов'язані з процесом забезпечення якості:

- **аналіз і планування;**
- **розробку тестових сценаріїв;**
- **оцінку критеріїв закінчення тестування;**
- **написання звітів;**
- **рецензування документації (в тому числі і вихідного коду);**
- **проведення статичного аналізу.**

Загальні принципи тестування

1. **Тестування показує наявність дефектів,** але не дозволяє довести їх відсутність.
2. **Повне тестування неможливо** - неможливо провести вичерпне тестування, за виключенням зовсім примітивних випадків.
3. **Раннє тестування** - починати якомога раніше в життєвому циклі розробки ПЗ, зусилля тестування повинні бути сконцентровані на певних цілях.
4. **Накопичення дефектів** - щільність скупчення дефектів в різних елементах програми може відрізнятися. Принцип Парето:

80% проблем містяться в 20% модулів

Загальні принципи тестування

5. Парадокс пестициду – виконуючи одні і ті ж тести знову і знову - вони знаходять все менше нових помилок. Необхідно періодично вносити зміни в набори тестів, і коригувати їх

6. Тестування залежить від контексту - методологія, техніка і тип тестування повинні прямо залежить від природи самої програми.

Також:

- **тестування має проводитися незалежними фахівцями;**
- **необхідно тестувати як позитивні, так і негативні сценарії;**
- **не допускати змін в програмі в процесі тестування;**
- **вказувати очікуваний результат тестів.**

Класифікація дефектів

З точки зору впливу на працездатність ПЗ

1. Blocker – помилка, яка приводе ПЗ в неробочий стан. Подальша робота - **неможлива**.
2. Critical - приводить деякий ключовий функціонал в неробочий стан. Також - суттєве відхилення від бізнес логіки, **неправильна реалізація** необхідних функцій, втрата призначених для користувача даних і т.д.
3. Major - серйозна помилка, яка свідчить про відхилення від бізнес логіки або **порушує роботу** ПЗ. **Не має критичного впливу** на додаток.
4. Minor - **не порушує** функціонал тестованого ПЗ, але є невідповідності результату (дизайн).
5. Trivial - **не впливає** на функціонал.

Класифікація дефектів

З точки зору пріоритетності виправлення

1. High – баг повинен бути виправлений якомога швидше, тому що він критично впливає на працездатність програми.
2. Medium - дефект повинен бути обов'язково виправлений, але він не робить критичний вплив на роботу програми.
3. Low - помилка повинна бути виправлена, але вона не має критичного впливу на програму і усунення може бути відкладено, в залежності від наявності інших більш пріоритетних дефектів.

Життєвий цикл дефекту

1. New – Тестувальник знайшов баг, дефект занесено в «Bug-tracking» систему.
2. Opened. Project manager аналізує дефект вирішує → наступний стан:
   - Deferred – відкладено. Виправлення цього бага не несе цінності на даному етапі розробки або по іншим, відстрочує його виправлення причин.
   - Rejected - відхилена. Дефект не рахується дефектом або вважатися неактуальним.
   - Duplicate - помилка вже раніше була внесена в «Bug-tracking» систему.

Життєвий цикл дефекту

3. Assigned - призначено. Якщо помилка актуальна і повинна бути виправлена в наступній збірці, відбувається призначення на розробника який повинен виправити помилку.
4. Fixed - виправлено. Розробник заявляє, що усунув баг. Залежно від того, чи виправив розробник дефект отримує наступні стани:
   - Verified - перевірено. Даний статус баг отримує після перевірки тестувальником в наступному релізі чи дійсно розробник виправив дефект.
   - Reopened - повторно відкритий. Якщо баг в новому релізі не виправлено.

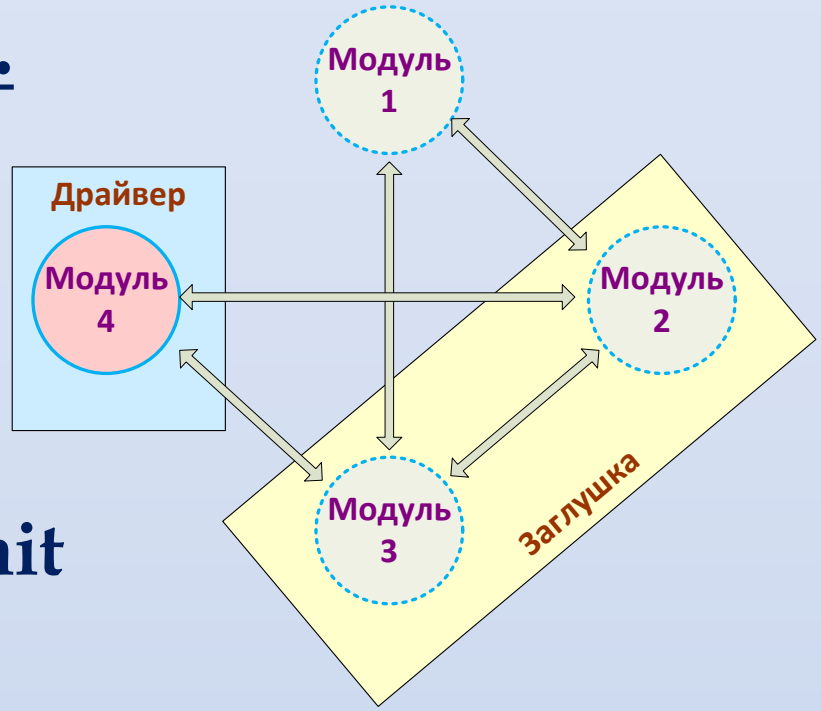
Життєвий цикл дефекту

5. <u>Closed</u> – баг остаточно усунуто.

Рівні тестування.

1. Модульне тестування.

(*Unit testing*) - тестування кожної функціональності програми окремо, в штучно створеному середовищі.



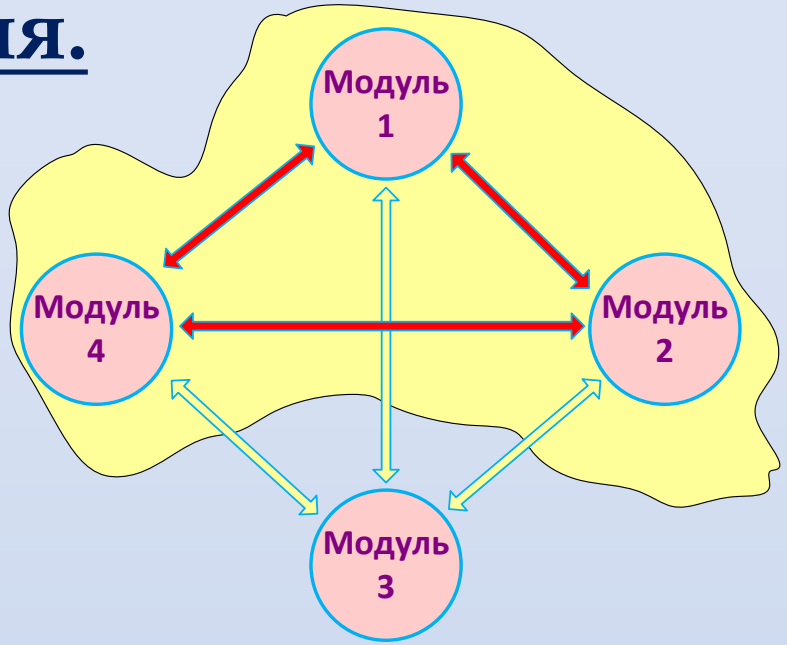
Середовище для деякого unit створюється за допомогою драйверів і заглушок.

Драйвер - певний модуль тесту, який виконують тестований нами елемент.
Заглушка - частина програми, яка симулює обмін даними з тестованим компонентом, виконує імітацію робочої системи.

Рівні тестування.

2. Інтеграційне тестування.

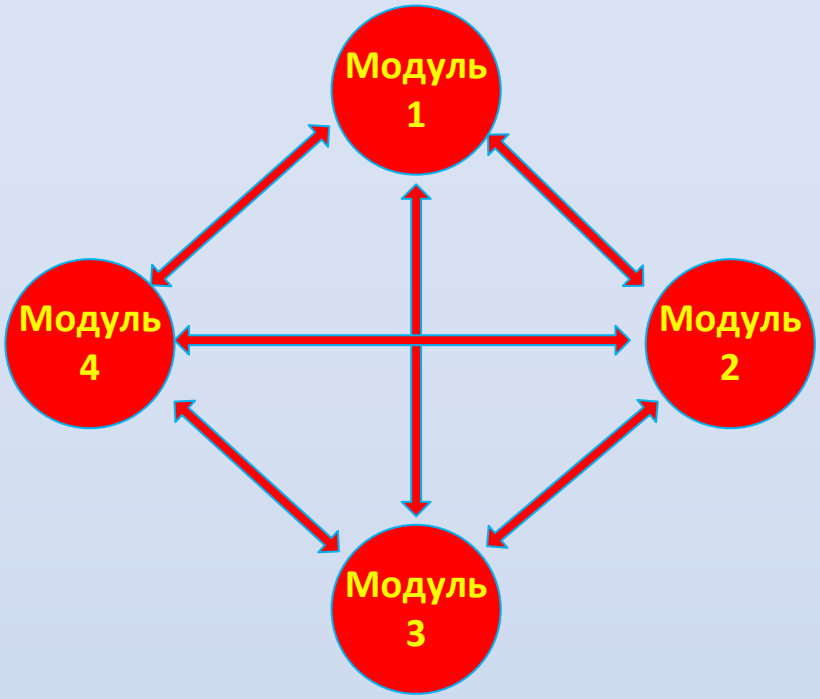
Тестування, при якому на відповідність вимог перевіряється інтеграція модулів, їх взаємодія між собою, а також інтеграція підсистем в одну загальну систему.



Використовуються компоненти, вже перевірені за допомогою модульного тестування, які групуються (множина).
Множина перевіряються відповідно до плану тестування, складеним для них, а об'єднуються вони через свої інтерфейси.

Рівні тестування.

3. Системне тестування.



- тестування ПЗ виконується з метою перевірки відповідності системи вихідним вимогам, як функціональним, так і не функціональним.

Дефекти, що виявляються:

- **Неправильне використання системних ресурсів.**
- **Непередбачені комбінації призначених для користувача даних.**
- **Проблеми з сумісністю оточення.**
- **Непередбачені сценарії використання.**
- **Невідповідність з функціональними вимогами.**
- **Погана зручність використання.**

Рівні тестування.

4. Приймальне тестування.

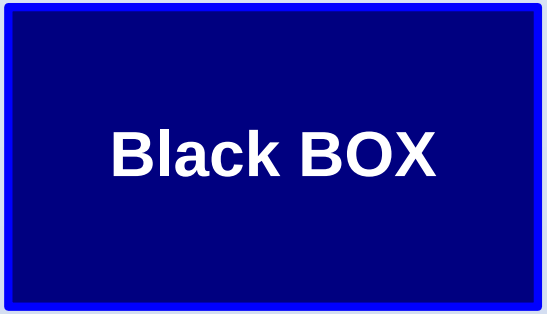
Метою *acceptance* тестування є визначення готовності продукту, що досягається шляхом проходу тестових сценаріїв і випадків, які побудовані на основі специфікації вимог щодо розроблюваного ПЗ.
Проводиться на етапі здачі готового продукту замовнику.
Результатом приймального тестування може стати:

- **Відправлення проекту на доопрацювання.**
- **Ухвалення його замовником, як виконаного завдання.**

Типи тестування

1. White/Black/Grey Box-тестування

Техніка тестування, заснована на роботі виключно з зовнішніми інтерфейсами системи.



Метою тестування технікою Black Box є пошук:
- **неправильно реалізовані або відсутні функції;**
- **помилок інтерфейсу;**
- **помилок в структурах даних або організації доступу до зовнішніх баз даних;**
- **помилок поведінки або недостатня продуктивність системи;**

Типи тестування

2. White/Black/Grey Box - тестування

Техніка білого (прозорого, відкритого, скляного ящика) тестування передбачає, що внутрішня структура / пристрій / реалізація системи відомі.

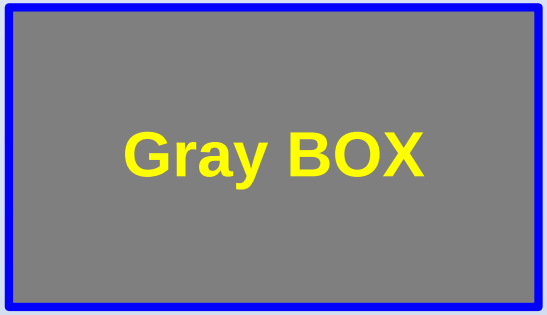
Знання всіх особливостей програми, що тестується і її реалізації - обов'язкові для цієї техніки. Поглиблення під внутрішній устрій системи, за межі її зовнішніх інтерфейсів.

При цьому процедура написання або вибору тест-кейсів на виконується на основі аналізу внутрішнього устрою системи або компонента.

Типи тестування

3. White/Black/Grey Box - тестування

Техніка тестування, заснована на комбінації White Box і Black Box підходів.



Внутрішній устрій програми нам відомо лише частково.
Передбачається доступ до внутрішньої структурі та алгоритмам роботи ПЗ для написання максимально ефективних тест-кейсів, але саме тестування проводиться за допомогою техніки чорного ящика, тобто, з позиції користувача.

Види тестування

1. Функціональне тестування - спрямовано на перевірку відповідності реальних характеристик ПЗ до визначених SRS функціональних вимог. Завдання функціонального тестування – підтвердити, що розроблене ПЗ володіє всім функціоналом, необхідним замовником.

При цьому для тестування створюються тестові випадки (***testcases***), складання яких враховує пріоритетність функцій ПО, які необхідно покрити тестами.
Таким чином можна переконатися в тому, що всі функції розроблюваного продукту працюють коректно при різних типах вхідних даних, їх комбінацій, кількості і т.д.

Види тестування

2. <u>Нефункціональне тестування</u> – тестування властивостей, які не належать до функціональності системи.

- **Надійність -** реакція системи на непередбачені ситуації.
- **Продуктивність -** працездатність системи під різними навантаженнями.
- **Зручність -** дослідження зручності роботи з додатком з точки зору користувача.
- **Масштабованість -** вимоги до горизонтального або вертикального масштабування додатку.
- **Безпека -** захищеність призначених для користувача даних.
- **Мобільність -** переносимість додатку на різні платформи.

Види тестування

3. Регресійне тестування – тестування спрямоване на виявлення так званих регресійних помилок у вже протестованих компонентах ПЗ.

Регресивні помилки - ті ж баги, що з'являються не при написанні програми, а при додаванні в існуючий реліз нових компонентів програми або виправлення інших багів, що може стати причиною виникнення нових дефектів в уже протестованому продукті.

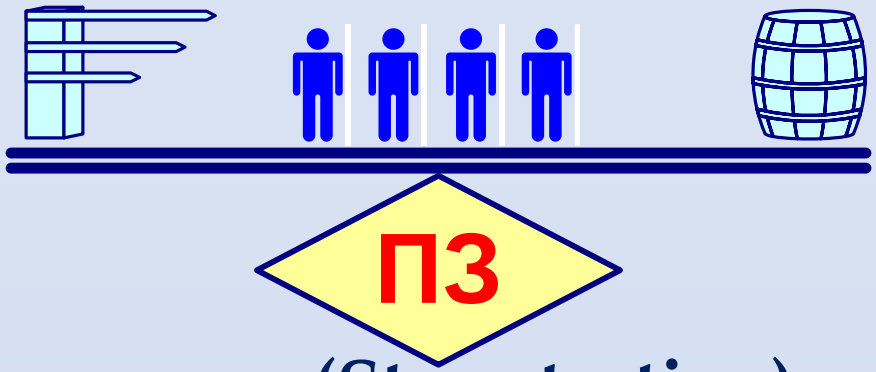
Мета регресійного тестування - переконатися, що виправлення одних багів не спричиняє виникнення інших і що оновлення ПЗ не створила нових дефектів в уже перевіреному коді.

Види тестування

4. Тестування продуктивності – оцінка робочих можливостей, стабільності, обсяг ресурсів, необхідних для використання в умовах різних сценаріїв та навантаження.

- **Тестування навантаження (Loadtesting) - тестування залежності часу відклику для запитів різних типів, з метою підтвердження, що ПЗ працює у відповідність до вимог при звичайному навантаженні.**

Види тестування



- **Стрес - тестування (Stresstesting) - тестування робочих можливостей при застосуванні навантаження, що перевищує типові в декількох разів.**
- **Тестування стабільності або напрацювання на відмову (Stability/Reliabilitytestin) - перевіряє робочу функціональність при тривалій роботі часу, при нормальній програмі навантаження.**
- **Об'ємне тестування (VolumeTesting) - тестування проводиться із збільшенням кількість використаних даних, які зберігаються і використовуються в додатку..**

Документування

1. **Test case – перелік, опис та послідовність дій та (або) умов, необхідних для перевірки певної функціональності ПЗ, що розробляється.**

Включає:
Action – дія, що виконується.
Expected result очікуваний результат
Test result - фактично отриманий результат.

	A	B	C	D	E	F	G	H	I	J
1	Test Case ID		BU_001	Test Case Description		Test the Login Functionality in Banking				
2	Created By		Mark	Reviewed By		Bill		Version		2.1
3										
4	QA Tester's Log		Review comments from Bill incorprate in version 2.1							
5										
6	Tester's Name		Mark	Date Tested		1-Jan-2017		Test Case (Pass/Fail/Not		Pass
7										
8	S #	Prerequisites:				S #	Test Data			
9	1	Access to Chrome Browser				1	Userid = mg12345			
10	2					2	Pass = df12@434c			
11	3					3				
12	4					4				
13										
14	Test Scenario	Verify on entering valid userid and password, the customer can login								
15										
16	Step #	Step Details		Expected Results		Actual Results			Pass / Fail / Not executed /	
17										
18	1	Navigate to http://demo.guru99.com		Site should open		As Expected			Pass	
19	2	Enter Userid & Password		Credential can be entered		As Expected			Pass	
20	3	Click Submit		Cutomer is logged in		As Expected			Pass	
21	4									
22										
23										

Документування

2. Bug report – технічний документ, який містить в собі повний опис бага, що включає інформацію, як про сам баг (короткий опис – *summary* , серйозність - *severity*, пріоритет - *priority* і т.д.), так і про умовах виникнення даного бага.

Баг репорт повинен містити правильну, єдину термінологію, що описує елементи призначеного для користувача інтерфейсу і події даних елементів, що призводять до виникнення бага.

Short bug description	Severity	Priority	Full bug description
Overflow	High	Medium	If set large values forsides we get wrong area calculation. For example: First side =999; Second side =999; Third side =999 Calculated Area = 15,2

Системи автоматизованого тестування

HP:

- **HP LoadRunner,**
- **HP QuickTest Professional,**
- **HP Quality Center**

IBM:

- **Rational FunctionalTester,**
- **Rational PerformanceTester,**
- **Rational TestStudio**

Segue SilkPerformer
SmartBear Software TestComplete

https://en.wikipedia.org/wiki/Test_automation

Рекомендована ЛІТЕРАТУРА

- **Томашевський О.М., Цегелік Г.Г., Вітер М.Б., Дудук В.Ш. Інформаційні технології та моделювання бізнес-процесів.** Навч. посіб. – К.: «Видавництво «Центр учбової літератури», 2012. – 296 с.
- **Карпенко М.Ю., Манакова Н.О., Гавриленко І.О. Технології створення програмних продуктів та інформаційних систем.** Навч. посіб. - Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім.О.М. Бекетова, 2017. – 93 с.
- **Алексенко О.В. Технології програмування та створення програмних продуктів.** Конспект лекцій. – Суми, Сумський державный університет, 2013. – 133с.

Рекомендована ЛІТЕРАТУРА

- **Jorgenson P.C. Software testing.** – NY.: CRC Press, 2014. – 437 p.
- **Vance S. Quality Code. Software Testing Principles, Practices and Patterns.** – NY.: Addison-Wesley, 2014. – 231 p.

Посилання

- **ISTQB – International Software Testing Qualification Board** https://www.istqb.org/
- **ISTQB – Glossary** https://glossary.istqb.org/en/search/

Контрольні запитання

- Надайте визначення дефектів програмного продукту та головні причини їх появи. Наведіть класифікацію дефектів з точки зору впливу на працездатність ПЗ та пріоритетності. Поясніть необхідність тестування ПЗ.
- Надайте визначення якості програмного продукту, перелік видів якості та пояснить модель якості ПЗ.
- Надайте визначення тестування ПЗ та вкажіть базові принципи тестування. Поясніть життєвий цикл дефекту.
- Надайте перелік рівнів тестування та пояснить їх сутність.
- Поясніть сутність типів тестування White/Black/Gray Box та видів тестування.

The END
Mod 2. Lec 5.