# Homework for 6 Beetroot lesson “Testing levels and types”

## Beet Seed

### Склади порівняльну таблицю функціонального, нефункціонального і пов’язаного зі змінами видів тестування, що відповідає на питання “Що перевіряється?”, “Коли застосовується?”, “Які обмеження та можливості?”

| **Вид тестування** | **Що перевіряється** | **Коли застосовується** | **Обмеження** | **Особливості** |
| --- | --- | --- | --- | --- |
| **Функціональні види тестування** | | | | |
| **Функціональне тестування** | Відповідність функцій ПЗ заданим вимогам | На етапах розробки, тестування, перед релізом | Не перевіряє продуктивність, зручність, безпеку | Виконується вручну чи автоматизовано, базується на специфікації |
| **Тестування користувацького інтерфейсу** | Правильність відображення елементів, зручність використання | При розробці та перевірці UI, перед релізом | Може пропустити проблеми функціоналу, які не залежать від UI | Перевіряє вигляд, поведінку елементів і реакцію системи відповідно інтерфейсу |
| **Тестування взаємодії** | Коректність взаємодії між модулями чи системами | Після інтеграції модулів чи компонентів | Не виявляє проблем всередині самих модулів | Фокусується на протоколах, API, передачі даних |
| **Нефункціональні види тестування** | | | | |
| **Навантажувальне тестування** | Поведінка системи при нормальних та підвищених навантаженнях | Перед релізом, у середовищах, де очікується високе навантаження | Вимагає специфічних інструментів, не виявляє багів функціоналу | Допомагає виявити вузькі місця у продуктивності |
| **Стрес-тестування** | Поведінка системи за межами допустимих навантажень | Перед релізом у критичних системах | Може не бути релевантним для всіх систем | Перевіряє, чи система не "зламається" у критичних умовах |
| **Здатність до відновлення** | Як система відновлює працездатність після збоїв | У критично важливих системах | Вимагає специфічних сценаріїв для створення збоїв | Перевіряє резервні копії, автоматичне відновлення |
| **Тестування стабільності або надійності** | Тривала робота системи без збоїв під стандартним навантаженням | Перед релізом, особливо для довготривалих процесів | Вимагає багато часу для проведення | Імітує роботу системи у реальному часі |
| **Об'ємне тестування** | Поведінка системи з великим обсягом даних | У базах даних, системах, де працюють із великими файлами чи потоками даних | Вимагає великих ресурсів для моделювання | Перевіряє швидкість роботи, можливість обробки великого обсягу |
| **Тестування встановлення** | Чи правильно встановлюється ПЗ, коректність налаштування | Перед релізом, після змін в інсталяційних процесах | Не охоплює всі аспекти роботи ПЗ після встановлення | Включає перевірку встановлення, оновлення |
| **Конфігураційне тестування** | Робота системи в різних конфігураціях середовища | Після розробки, перед релізом | Не охоплює перевірку функціоналу | Допомагає уникнути конфліктів із зовнішнім ПЗ чи пристроями |
| **Тестування сумісності** | Коректна робота з іншими системами чи платформами | Перед релізом у багатоплатформних системах. | Вимагає багато середовищ для перевірки. | Перевірка з різними ОС, браузерами, обладнанням, налаштуваннями |
| **Тестування зручності використання** | Зручність інтерфейсу, чи інтуїтивно зрозумілий користувачам | Перед релізом, після змін в UI/UX | Суб'єктивність оцінки, залежність від вибраної аудиторії | Може включати опитування реальних користувачів та результати alpha\beta тестування |
| **Тестування безпеки** | Захищеність системи від атак, витоків даних | У критично важливих системах перед релізом | Вимагає специфічних інструментів та знань у сфері безпеки | Включає перевірку авторизації, автентифікації, шифрування даних |
| **Тестування пов'язане зі змінами** | | | | |
| **Димове тестування** | Основна працездатність ключових функцій | На початку тестування нової збірки | Не охоплює детальну перевірку | Швидке, поверхневе тестування |
| **Регресійне тестування** | Чи не вплинули зміни на вже існуючий функціонал | Після внесення змін у код | Вимагає багато часу для комплексної перевірки | Може бути автоматизованим для повторних запусків |
| **Повторне тестування** | Перевірка виправлених дефектів | Після фіксації багів | Не охоплює перевірку нових дефектів | Виконується на основі вже знайдених дефектів |
| **Тестування збірки** | Чи правильно інтегровані модулі та чи коректно вони взаємодіють в новій збірці | На етапі інтеграції | Може пропустити окремі проблеми модулів | Перевіряє коректність побудови системи |
| **Санітарне тестування / Перевірка справності** | Основна працездатність після внесення мінімальних змін | Після дрібних змін чи апдейтів | Обмежене за масштабом перевірок | Швидкий і вибірковий тест |

### Поясни, в чому різниця між регресією та ретестингом (5 речень)

Взагалі-то я вважаю, що обидва ці підходи доповнюють одне одного: ретестинг підтверджує, що баги виправлено, а регресія гарантує, що ці виправлення не зламали інші частини системи.

Але якщо говорити про різницю між ними, це можна виділити в окремі порівняльні тезиси:

* **Регресійне тестування** перевіряє, **чи не вплинули зміни в коді** на раніше працюючий функціонал, тоді як **ретестинг** **фокусується** лише **на перевірці виправлених дефектів**;
* **Регресія** **охоплює ширший набір тестів** для перевірки загальної стабільності, тоді як **ретестинг повторно виконує конкретні тести** для дефектів;
* **Регресія часто автоматизується**, щоб перевірити великий обсяг функціоналу, а **ретестинг здебільшого виконується вручну**, особливо якщо баги виправлялися недавно;
* **Ретестинг не займається пошуком нових дефектів, на відміну від регресії,** яка може виявити проблеми, спричинені змінами в інших частинах системи.

## Beet Sprout

### Як ти вважаєш, чи можливе для продукту проведення тільки функціонального тестування, без перевірки нефункціональних вимог?

Я вважаю, що проведення фунціонального тестування без перевірки нефункціональних вимог можливо, але це непрофесійно та не є доречним, адже значно знижує контроль якості та надійність продукту. І ось чому:

По-перше, функціональне тестування перевіряє лише відповідність функцій вимогам, але не гарантує, що продукт працює ефективно, надійно або зручно в реальних умовах - а отже, таке тестування не враховує ані суб’єктивні оцінки користувачів, ані реальні фактори, що можуть критично повпливати на роботу в продакшені.

По-друге, якщо повертатись до ризиків “несуб’єктивності” тестування та важливих факторів, проведення тестування без урахування нефункціональних вимог може призвести до недостатньої перевірки зручності та безпеки (невдоволення користувачів та великий ризик втрати або перехвату даних), підвищеному ризику у виробництві та експлотуванні (проблеми стабільності, сумісності або конфігурації), або ж навіть до фінансових втрат.

І все ж таки, по-третє, відповідь більше залежить від типу продукту: у простих програмних продуктах (наприклад, калькуляторах чи невеликих утилітах) нефункціональні вимоги можуть бути менш важливими, а отже їх можна опустити задля економії часу перед релізом, наприклад. Але для масштабних систем або веб-додатків лише функціональне тестування не є достатнім для високого забезпечення якості.

### Як ти розумієш необхідність проведення smoke (димового) тестування? Чи завжди воно є доречним?

На мій погляд, необхідність проведення smoke-тестування визначається його ключовою метою — перевірити базову працездатність основних функцій системи після нової збірки.

Таке тестування доречне, наприклад, коли ПЗ, що розробляється, є модульною системою, а в якості методології розробки використовується Agile підхід. Припустимо, що вийшло оновлення для модулю аутентифікації\реєстрації в системі загального державного облікового реєстру громадян - я вважаю, що саме в такому випадку таке поверхневе тестування є ключовим для загальної оцінки працездатності конкретного модулю.

Якщо ж ПЗ є монолітною системою, яку продовжують підтримувати протягом вже 10 років, і для неї вийшло оновлення інтерфейсу - не думаю, що саме smoke-тестування було б хорошим варіантом : )

А ось, наприклад, проведення спочатку регресивнного, а після ще й ретестингу було б чудовим вибором для такого кейсу.