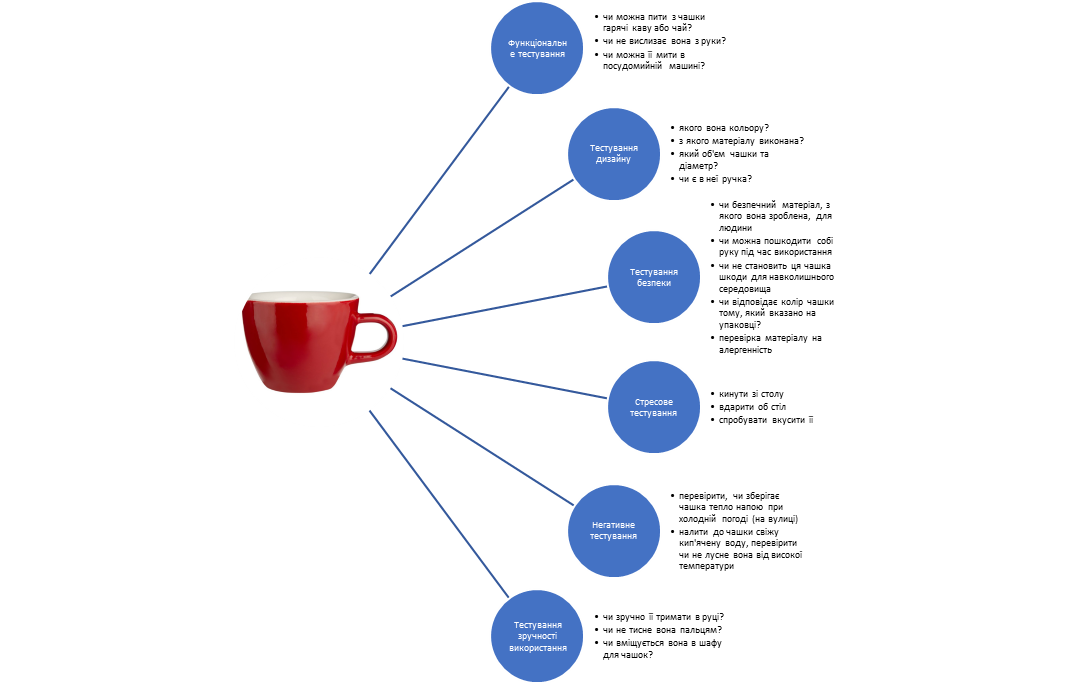
**Перший рівень. 1.**



Для того, щоб протестувати чашку, необхідно спочатку зрозуміти, яку саме чашку треба протестувати (уточнити вимоги). Отже, необхідно встановити об’єкт тестування, дізнатися його якісні характеристики та уточнити призначення.

Окрім того, треба розповісти якими способами ми будемо тестувати цей предмет.

Тестування чашки також залежить від середовища, де ми будемо його виконувати. Важливо зрозуміти, хто кінцевий користувач чашки, та де її будуть використовувати.

Також треба вияснити, чи є дедлайни та які є критерії оцінки. А далі - необхідно переходити до тестування за різними напрямами (як показано на схемі).

**2. Верифікація** - процес, під час якого ми виясняємо, чи відповідає наш продукт заданим вимогам. Це може бути перегляд документації, дизайну, коду та програми, щоб перевірити, чи було ПЗ створено у відповідності до початкових вимог.

**Валідація -** тут ми розуміємо, чи відповідає продукт потребам кінцевого користувача. Валідація має гарантувати, що ПЗ виконує бажані функції у своєму середовищі.

* процес верифікації включає перевірку документації, дизайну, коду та програми, а валідація включає в себе тестування і перевірку самого продукту;
* верифікація - відповідність ПЗ специфікації, а валідація - відповідність ПЗ вимогам та очікуванням;
* замовника зазвичай цікавить в більшій мірі валідація (задоволення власних потреб), а виконавця - як дотримання всіх норм якості та вимог (верифікація), так і відповідність всіх особливостей продукту бажанням замовника.
* 

**Другий рівень.**

|  | **Плюси** | **Мінуси** |
| --- | --- | --- |
| **Product** | * розробники працюють над одним продуктом та відчувають свою приналежність до створення нового, свій вклад в продукт; * велика залученість в розробку та малий бар’єр між розробником та клієнтом дозволяють девелоперам перетворюватися з простих виконавців в свідомих інженерів, які розуміють бізнес задачі; * в продуктах часто більш сильні команди, вони будують більш довгострокові стосунки з працівниками (бо націлені на результат); * у розробників є більше можливостей впливати на продукт, приймати рішення, брати на себе відповідальність; * можливість побачити зсередини роботу всіх функцій технологічної компанії (маркетинг, дизайн, розробка, підтримка і т.д.), перейняти цей досвід для власних проектів у майбутньому; * деякі компанії надають можливість отримати власну долю в бізнесі (stock options); * часто така компанія веде більш чітку політику зарплат: ставка співробітника залежить від його кваліфікації та досвіду, а не від того, на яку суму компанії вдалося домовитися із замовником | * в продуктові компанії складніше потрапити, є більш високі вимоги до спеціалістів; * відсутність можливості міграції між проектами і техологіями всередині компанії; * є ризик, що продукт не стане успішним, особливо на стадії стартапу; * багато продуктових компаній очікує від співробітника роботи з повної віддачею - овертайми; * часто існує певна консервативність в прийнятті рішень, виборі інструментів розробки та запровадженні модних трендів |
| **Outstaff** | * відсутній конфлікт інтересів між підрядником та замовником щодо зарплат - зарплата розробників є прозорою для замовника проекту; * близькість розробника до замовника, можна якісніше обговорювати деталі рішень безпосередньо в процесі розробки; * персональний підхід - замовник обирає кожного спеціаліста особисто | * часто відсутні джуніор-позиції, адже клієнт хоче краще заплатити більше, але взят на роботу більш кваліфікованого співробітнка; * складніше перейти на інший проект, адже розробники тісніше пов’язані з замовником; * робота на двох “босів” - іноземного замовника та місцевого посередника |
| **Outsource** | * швидке освоння різних технологій, * розширення зони інтересів, працюючи в різних напрямах; * більш швидке кар’єрне зростання; * відносно легше влаштуватися на роботу (часто є “текучка кадров”); * можна швидко навчитися не лише новим технологіям, а й освоїти такі важливі навички як тайм-менеджмент, організація своєї роботи, багатозадачність. | * нема відчуття “product ownership”; * небезпека затриматися на нецікавому довгостроковому проекті; * нема відчуття корпоративних цінностей компанії (не завжди); * проблема комунікації: складна ієрархія, затримка швидкого прийняття рішень; * високе навантаження - аутсорсингові компанії часто конкурують одна з одною за клієнтів і намагаються отримати якнайбільше проектів, закриваючи їх якнайшвидше + погана комунікація на різних рівнях менеджменту. Співробітники можуть працювати багато та інтенсивно; * нестабільність - такі компанії часто залежать від клієнтів і проект може закінчитися, а розробників можуть звільнити. |

**Приклади невдалої валідації та верифікації з життя**

* невдала верифікація - у автомобілі не працює педаль газу (наявність та справна робота такої педалі визначена вимогами до авто), а невдала валідація - кнопки збільшення гучності музики в цьому авто знаходяться в незручному для водія місці, під час руху неможливо додати гучності (користувачі очікували, що ці кнопки будуть в зручному місці)
* невдала верифікація - з мобільного телефону не можна подзвонити (це основна функція телефону, яка передбачена ТЗ), невдала валідація - нова версія цього телефону занадто яскравого кольору (споживачі очікували більш стильного дизайну).

**Третій рівень**

**Важливість принципів тестування:**

1. Раннє тестування заощаджує час і гроші.

Тестування повинне починатися якомога раніше в життєвому циклі розробки ПЗ. Чим швидше в життєвому циклі ми знаходимо баг, тим простіше і дешевше його пофіксити. Звичайне тестування є набагато дешевшим, ніж тестування прямо перед релізом. Після того, як остаточні вимоги визначені, тестувальників можна залучти до тестування. Тобто тестування слід проводити на етапі вимог, щоб навіть якщо вимоги будуть визначені неправильно, їх можна було негайно виправити, а не фіксувати вже на стадії розробки

1. Вичерпне тестування неможливе.

Неможливо перевірити всі можливості пристрою з усіма можливими комбінаціями під час тестування. Замість цього можнаа розглянути тестування кількох комбінацій на основі пріоритету з використанням різних методів. Повне вичерпне тестування вимагає багато зусиль, і більшість з них є неефективними. До того ж, часові рамки проекту не дозволять протестувати настільки велику кількість комбінацій. Наприклад, якщо в програмі є поле, куди можна вводити букви, цифри та спеціальні символи, то кількість комбінацій для тестування є просто величезною.

1. Тестування залежить від контексту.

На ринку є багато напрямів, де застосовується тестування. Це може бути банківська справа, страхування, медицина, подорожі, реклама. Тому кожен з цих доменів має різні вимоги, функції, мету тестування, ризики, методи і т.д. Наприклад, тестування банківської програми відрізняється від тестування застосунку для замовлення товарів, або рекламного сайту. Тому неефективно використовувати один і той же метод для різних продуктів.

