# Homework for 3 Beetroot lesson “DLC and STLC. Software development methodologies”

## Beet Seed

### Склади порівняльну таблицю найбільш поширених методологій

| **Методологія** | **Основні риси** | **Етапи** | **Переваги** | **Недоліки** | **До якої галузі доречна** |
| --- | --- | --- | --- | --- | --- |
| **Waterfall** | Лінійна послідовна модель, кожен етап завершується перед початком наступного. | Аналіз →  Дизайн → Реалізація → Тестування → Впровадження → Обслуговування | Чітка структура, легке управління, простота: підходить для малих проєктів із чіткими вимогами. | Немає гнучкості, складно вносити зміни після завершення етапів, високі ризики для великих проєктів. | Державні проєкти, системи з фіксованими вимогами.  Наприклад: державний реєстр, транспортні системи або невеликі прототипи ПЗ для остаточного прийняття рішення про розробку в майбутньому. |
| **V-model** | Розширення Waterfall із додаванням процесів тестування на кожному етапі. | Аналіз →  Дизайн → Реалізація → Верифікація → Валідація → Впровадження | Висока якість продукту завдяки інтеграції тестування, добре підходить для критичних систем. | Складно адаптувати до змін, дорога модель для невеликих проєктів. | Критичні та відмовостійкі системи.  Наприклад: ПЗ для космічних апаратів, звичайна авіація, медицина. |
| **Spiral Model** | Ітеративна модель із фокусом на аналіз ризиків. | Планування → Аналіз ризиків → Реалізація → Оцінка → Повторення (ітерації) | Підходить для великих проєктів, можливість коригування вимог на кожній ітерації, зниження ризиків. | Складність управління, високі витрати, вимагає досвідчених спеціалістів. | Великі довготривалі проєкти.  Наприклад: розробка ПЗ для фінансової галузі або мережевого обладнання. |
| **Kanban** | Гнучкий підхід із візуалізацією роботи на дошках (Kanban Board). | Постійне планування → Виконання → Контроль → Завершення | Простота реалізації, прозорість процесів, підходить для підтримки та проєктів із гнучкими вимогами. | Може виникнути хаос без чіткого контролю, не підходить для проєктів із жорсткими строками. | IT-підтримка, DevOps, невеликі гнучкі команди.  Наприклад: сучасний веб-застосунок, галузь будівництва. |

### Надаючи відповіді, обґрунтуй свою думку: чому це саме сильна/слабка сторона/доцільна галузь застосування

#### Waterfall:

* **Сильні сторони:**
  + **Чітка структура:** Кожен етап має чіткий початок і кінець, що робить цю модель передбачуваною. Це важливо для державних проєктів, де узгодження з замовником і регулятором займає значний час.
  + **Документованість:** У державних проєктах потрібна велика кількість документів, які Waterfall допомагає створювати паралельно з процесом розробки.
* **Слабкі сторони:**
  + **Відсутність гнучкості:** Будь-які зміни в пізніх етапах потребують повернення до попередніх, що дорого і складно. Наприклад, якщо змінюються вимоги до державної інформаційної системи після розробки дизайну, це призводить до перевитрати ресурсів.
* **Доцільна галузь застосування:**
  + Для державних проєктів, таких як реєстри або транспортні шляхи залізниці, тому що вимоги зазвичай затверджуються до початку розробки. Внесення змін у процесі розробки та впровадження таких проєктів зазвичай несе величезні затрати та може призвести до повного реконструювання ідеї, а отже є майже неможливим.

#### V-model:

* **Сильні сторони:**
  + **Підвищена якість через тестування:** Тестування на кожному етапі дозволяє виявити помилки раніше. Для галузей, де ціна помилки висока - це життєво необхідно, адже помилки можуть коштувати навіть втратою життя людей.
* **Слабкі сторони:**
  + **Висока вартість:** Постійне тестування та перевірки потребують більше ресурсів і часу. Наприклад, у невеликому стартапі це буде фінансово невигідно та недоцільно.
* **Доцільна галузь застосування:**
  + У системах критичної інфраструктури, таких як авіаційне ПЗ або медичні пристрої, важливо мінімізувати ризики через постійне тестування.

#### Спіральна модель:

* **Сильні сторони:**
  + **Гнучкість та аналіз ризиків:** Можливість адаптувати вимоги на кожному ітераційному циклі. Це ідеально для великих проєктів із невизначеними початковими вимогами. Наприклад, у розробці фінансового ПЗ часто змінюються нормативні вимоги.
* **Слабкі сторони:**
  + **Складність і вартість:** Вимагає багато ресурсів для аналізу ризиків, що робить її менш доцільною для проєктів із фіксованими бюджетами чи строками.
* **Доцільна галузь застосування:**
  + Великі й довготривалі проєкти, наприклад як розробка банківських систем, зазвичай вимагають мінімізацію невизначеності і, в подальшому, помилок.

#### Канбан:

* **Сильні сторони:**
  + **Простота та гнучкість:** Підходить для проєктів із постійно змінними вимогами, як наприклад, технічна підтримка або DevOps, де задачі можуть додаватися щодня.
* **Слабкі сторони:**
  + **Можливість хаосу:** Без чіткого контролю (наприклад, людський фактор) пріоритетів команда може зосередитися на менш важливих задачах, що знижує ефективність.
* **Доцільна галузь застосування:**
  + Технічна підтримка або робота невеликих гнучких команд, наприклад, у стартапах, де пріоритети можуть змінюватися миттєво, а швидке реагування на них є критично важливим.

## Напиши розгорнуті відповіді (0,5 - 1 сторінки тексту) на два питання

### На твою думку, чому з’явився Agile-маніфест?

Перш за все, я вважаю що маніфест методології мав дві ключові цілі: впровадити новий метод управління розробкою для будь-яких проєктів із зручними та зрозумілими правилами для всіх (на заміну застарілих методів з жорсткими методиками розробки, що у часи появи Agile явно проглядались у світі), а також надати чіткий та структурований опис цього методу для швидкого його застосування у нових та вже наявних проєктах різних масштабів.

По-друге, необхідність нового підходу для управління проєктів була на піку своєї необхідності: жорсткі процеси інших методологій обмежували творчість і ефективність команд, а також призводили до проблем в самих командах, адже проєкти почали вимагати постійної комунікації з клієнтами та швидкої адаптації до змін. Старі ж моделі управління, не враховували таких вимог.

Agile-маніфест прагнув переосмислити підхід до розробки ПЗ, зробивши його більш гнучким, командним та клієнтоорієнтованим, дозволяючи командам створювати продукти, які відповідають реальним вимогам клієнтів.

### Які проблеми він мав вирішити і чи це вдалося?

Agile-маніфест пропонував вирішити конкретні проблеми у процесах розробки ПЗ та управління проєктами, такі як:

* Низька гнучкість процесу розробки;
* Затримки у постачанні\впровадженні продукту;
* Низька залученість та освідомленість клієнта про роботу проєкту;
* Неефективна комунікація різних ролей в командах;
* Складність прогнозування ризиків.

Я вважаю, що на кожну з проблем Agile вплинув по-різному. Розглянемо детальніше:

* **Низька гнучкість**

Гнучкість стала основною перевагою Agile. Більшість сучасних IT-компаній використовують ітеративні підходи (Scrum, Kanban), що дозволяють швидко реагувати на зміни - а отже, так - на цей аспект методика значно повпливала, усунувши недоліки.

* **Затримки з продуктом**

Ітеративний підхід значно скоротив час до першої поставки функціонального продукту. Проте для деяких клієнтів часті проміжні версії можуть створювати ілюзію готовності продукту, що впливатиме на подальший процес розробки або ж підтримки ПЗ.

* **Низька залученість клієнта**

Agile-маніфест пропонував постійне залучення клієнта до процесу, і згодом клієнт став активним учасником розробки, хоч це і може викликати ризик постійної зміни вимог з його боку.

* **Неефективна комунікація команди**

В Agile-командах комунікація стала ключовим аспектом. Використання щоденних зустрічей (Scrum-мітинги) і крос-функціональних команд забезпечило ефективну взаємодію.

* **Прогнозування ризиків**

Я вважаю, що Agile частково повпливав на цей аспект: він зменшує ризики через ітеративність, але це не завжди працює в умовах високої невизначеності (наприклад, у великих проєктах із складними інтеграціями).

У підсумку - Agile вирішив більшість ключових проблем, а Agile-маніфест спрацював на практиці, але як і завжди, для кожного проєкту її команди в цій методології будуть свої плюси та мінуси. Головне знати чого хочеш та розуміти як реалізувати : )