**Перший рівень** — відпрацюй навички на базовому рівні.

Склади порівняльну таблицю найбільш поширених методологій:*а. Надаючи відповіді, обґрунтуй свою думку: чому це саме сильна/слабка сторона/доцільна галузь застосування.*

| **№** | **Назва методології** | **Сильні сторони** | **Слабкі сторони** |
| --- | --- | --- | --- |
| 1 | **WATERFALL**  Використовується переважно для розробки корпоративних додатків, таких як системи управління людськими ресурсами (HRMS), системи управління ланцюгами поставок, системи управління взаємовідносинами з клієнтами (CRM), системи управління запасами, точки продажу (POS) для роздрібних мереж тощо. Модель також була надзвичайно популярною в розробці програмного забезпечення. | * Вимоги сформульовані чітко і ясно, вони залишаються незмінними протягом усього розвитку проекту; * Детальне документування кожного етапу розробки забезпечує стійкість до змін у людських ресурсах - новий розробник може швидко отримати всю необхідну інформацію; * Ретельне планування структури розробки проекту зменшує кількість проблемних питань; * Встановлюються початкова та кінцева точки для кожної фази, що дозволяє легко вимірювати прогрес; * Завдання залишаються максимально стабільними протягом усього процесу розробки; * Забезпечує легкий контроль і прозорість для замовника завдяки суворій системі звітності; * Дата виходу готового продукту, а також його кінцева вартість можуть бути розраховані ще до початку розробки. | * Всі вимоги повинні бути відомі до початку розробки, що значно затримує старт проекту; * Низький рівень гнучкості ускладнює внесення змін під час розробки, або навіть робить це повністю неможливим; * Існує потреба в суворому управлінні та регулярному моніторингу, щоб проект вкладався у встановлені терміни; * Клієнт не має можливості ознайомитися з системою заздалегідь, тому не бачить продукт до моменту його завершення; * Якщо в процесі розробки стане зрозуміло, що продукт не відповідає вимогам ринку, можливості для змін не буде. |
| 2 | **V-MODEL**    Використання V-моделі може підпадати під проекти, які не зосереджуються на зміні вимог, наприклад, у проектах, що не передбачають зміни вимог: коли замовник має дуже чіткі задокументовані вимоги. Військові проекти. Проекти, пов'язані з критично важливими місіями, наприклад, космічний шатл. | * Нескладне використання: Середовище є досить зручним для користувача і забезпечує надійну базу для розробки програмного забезпечення для невеликих розробників. Середовище проектування є відносно простим і знаходить відверте використання в індустрії розробки програмного забезпечення. * Економія часу: V модель проста у використанні, а пробні дії, такі як проектування, планування та розробка, відбуваються перед кодуванням програмного забезпечення, що економить час для зайнятих розробників, які перевантажені роботою протягом дня. * Проактивне відстеження помилок: Середовище V Model складається з методу проактивного відстеження помилок, який допомагає інженерам виправляти баги та інші системні помилки і створювати програмне забезпечення за менший проміжок часу, ніж це передбачено графіком. * Прямолінійний дизайн.Розробники V-моделі працюють над тим, щоб зробити цю модель простою для розуміння; тому вони зробили архітектуру простою, але високоефективною. | * Висока жорсткість: Модель V має високу жорсткість.Структура може здаватися простою, але виконання і методи досить складні порівняно з іншими середовищами проектування або моделями. * Обмежена гнучкість: V-модель не дуже гнучка. Вона не є елементарною, і новим розробникам потрібен час, щоб звикнути до неї. Архітектура проста у використанні, але має обмежену гнучкість з точки зору дизайну. * Масове оновлення: Процес розробки програмного забезпечення пов'язаний з неприємностями та помилками. Інженери роблять все можливе, щоб розробити модель для створення найкращого програмного забезпечення, яке відповідає всім вимогам. Щоб переконатися, що розроблена модель є найкращою, потрібно пройти багато етапів спроб і помилок. * Ризиковано. Управління загальним середовищем є нестабільним і ризикованим. Воно не підходить для створення об'єктно-орієнтованого програмного забезпечення через невизначеності в дизайні. |
| 3 | **ІТЕРАТИВНА/СПІРАЛЬНА - AGILE**  Компанія Apple випустила кілька різних версій кожної моделі iPod. Наприклад, Nano еволюціонував від оригінального тонкого mp3-плеєра з дуже маленьким екраном до крихітної квадратної версії з сенсорним екраном, а потім і до довшої, тоншої версії з сенсорним екраном. Він еволюціонував від пристрою, який міг відтворювати лише музичні файли у форматі mp3, до пристрою для тренувань, який має спеціальний додаток Nike Ap для відстеження тренувань, а також до пристрою, який відтворює музику. Кожна з цих різних версій - це різні ітерації одного і того ж продукту. Мета - знайти ідеальну комбінацію функцій, яка сподобається клієнтам і надасть бажані послуги в правильному пакеті.  Корпорація Майкрософт випустила кілька версій операційної системи Windows, щоб додати функціональності для користувачів і виправити помилки. Кожна окрема версія - це окрема ітерація операційної системи, і мета полягає в тому, щоб наступна ітерація була кращою за попередню. | * Деяку робочу функціональність можна розробити і на початку життєвого циклу розробки програмного забезпечення . * Він легко адаптується до постійно мінливих потреб проекту та клієнта. * Він найкраще підходить для гнучких організацій. * Зміна обсягу або вимог в ітеративній моделі є більш економічно ефективною. * Можна планувати паралельну розробку. * Тестування та налагодження під час менших ітерацій є простим. * Ризики виявляються і вирішуються під час ітерації; і кожною ітерацією легко керувати. * В ітеративній моделі менше часу витрачається на документування і більше часу приділяється проектуванню. * Можна отримати надійний зворотній зв'язок з користувачами, представляючи ескізи та креслення продукту користувачам для зворотного зв'язку. | * Може знадобитися більше ресурсів. * Хоча вартість змін менша, але вона не дуже підходить для зміни вимог. * Потрібно більше уваги з боку керівництва. * Не підходить для невеликих проектів. * Для аналізу навичок потрібні висококваліфіковані ресурси. * Прогрес проекту сильно залежить від етапу аналізу ризиків. |
| 4 | **EXTREME PROGRAMMING**  Це гнучка методологія, створена наприкінці 90-х років для розробки програмного забезпечення для створення високоякісних систем, заснованих на тісній взаємодії з клієнтами, безперервному тестуванні та коротких циклах розробки. | * Mетодологія дозволяє компаніям, які розробляють програмне забезпечення, заощаджувати витрати і час, необхідний для реалізації проекту. Економія часу досягається завдяки тому, що XP фокусується на своєчасній доставці кінцевих продуктів. Зазвичай вони вирішують проблеми шляхом обговорення всередині команди. * Простота - ще одна перевага проектів Extreme Programming. Розробники, які вважають за краще використовувати цю методологію, створюють надзвичайно простий код, який можна вдосконалити в будь-який момент. * Весь процес в XP є видимим і підзвітним. * XP допомагає створювати програмне забезпечення швидше завдяки регулярному тестуванню на етапі розробки. * Екстремальне програмування сприяє підвищенню задоволеності та утриманню працівників. | * Eкстремальне програмування зосереджене на коді, а не на дизайні. Це може бути проблемою, тому що хороший дизайн надзвичайно важливий для програмних додатків. * Hе завжди добре ведеться документація дефектів. Відсутність документації дефектів може призвести до виникнення подібних помилок у майбутньому. * Не вимірює забезпечення якості коду. Це може призвести до дефектів у початковому коді. * XP - не найкращий варіант, якщо програмісти географічно роз'єднані. |
| 5 | **TEST-DRIVEN DEVELOPMENT** | * Гарантує, що всі коди добре протестовані. Ключовою характеристикою тесту є те, що він може не спрацювати, і команда розробників контролює Це принесе високу впевненість, коли ми випустимо програмне забезпечення у виробництво. * Менше коду. Вам потрібно писати лише мінімальні рядки коду, які пройдуть тест. Це допоможе зменшити дублювання коду, що дозволить швидше впроваджувати інновації та забезпечувати безперервну доставку. * Легше підтримувати. TDD робить ваш код гнучким і розширюваним. Код можна рефакторити або переміщати з мінімальним ризиком порушення коду. Оскільки рефакторинг є важливим кроком, який повторюється в кожній ітерації циклу TDD, код буде постійно підтримуватися. | * Це вимагає більше часу на написання тесту. Оскільки тест має бути заздалегідь визначений, нашим розробникам потрібен час для написання тестових кейсів, тому час розробки може бути довшим, ніж зазвичай. * Це вимагає більш досвідчених розробників. Це означає, що розробники, які працюють з TDD, повинні мати розуміння цієї техніки, володіти достатніми навичками для написання тестів, що не спрацьовують, а також вміти чітко слідувати процесу. |

**Другий рівень** — детальніше заглибся в практику.

1. Виконай завдання попереднього рівня.

2. Напиши розгорнуті відповіді (0,5 - 1 сторінки тексту) на такі два питання:

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

Якщо розібратись із хронологією виникнення різних методологій, то можна відстежити, що спочатку активно використовувались waterfall та ітеративні методології. Концепція Agile стала закономірним продовженням еволюції підходів, які використовувалися раніше. Історія показує, що більшого успіху досягають ті компанії, які комбінують різні інструменти, в залежності від специфіки проекту і контексту, в якому він реалізується.

В 20-му столітті люди стали свідками різкого розвитку науково-технічного прогресу і перед ними постало завдання більш досконалої систематизації та перегляд цінностей. Управлінці зрозуміли, що потрібно підвищити продуктивність своїх команд при нижчих затратах. З'явилася ідея розподілу завдань на блоки, щоденні стендапи (короткі планьорки не сідаючи), зворотний зв'язок користувачів та інші.

Наблизимось трохи ближче - 90-ті роки. Революція у сфері програмного забезпечення була неминучою, але все ж таки склалась ситуація, коли більшість проектів у гіршому випадку повністю провалювалися. Нарешті, на початку 2000-х років фактично народився Маніфест, де Agile опинився в центрі уваги. Чотири цінності та 12 принципів, які продовжують керувати підходом Agile, яким користуються команди і сьогодні. Ми маємо розуміти, що принципи Agile існували до створення Маніфесту. Люди застосовували їх для розробки програмного забезпечення і не тільки.

На мою думку, в недалекому майбутньому agile-практики також будуть еволюціонувати і в управлінні з’являться нові «революційні» ідеї.

* Які проблеми він мав вирішити і чи це вдалося?

Складається враження, що цей маніфест мав на меті вирішити всі проблеми, зважаючи на охоплення всіх принципів. Але ми все ж таки можемо виділити основні найбільш вагомі.

Виконавці часто стикались з тим, що порушували взяті на себе зобов'язання, їм був потрібен унікальний спосіб виправити це. Вони випускали на ринок неякісні продукти, недопрацьовані ПЗ. Потрібно було більше прозорості в тому, що відбувається. Люди потребували більше інформації про прогрес, який ми дійсно досягаємо в роботі над продуктом. Треба було швидше виводити продукти на ринок.

Комунікація завжди була одним із ключів до успіху і її теж треба було удосконалити. Актуальною темою завжди залишається фінансова спроможність у веденні проектів. Розробка програмного забезпечення коштує надто дорого, тому треба використовувати гнучкий підхід як спосіб знизити вартість продукту.

Навантаження невпинно зростало і часто спеціалістам і менджерам не вистачало ресурсів (технічних, людських тощо).

Я вважаю, що Agile Manifest доволі ефективно вирішує багато проблем, але його не можна розглядати як універсальну методологію. Будемо зважати також на те, що доволі важко імплементувати всі цінності в повному обсязі. Також не виключено, що скоро нам потрібне буде щось більш досконале.

Основна проблема, яку вирішує Agile-методологія, - це коли в проекті багато невідомих. Гнучкий підхід не є панацеєю від усіх бід. Agile дуже добре працює, коли потрібно зробити багато спроб і помилок, особливо це стосується інновацій та розробки продуктів.

**Третій рівень** — різнобічно опануй тематику уроку.

1. Виконай завдання двох попередніх рівнів.

2. Ти – засновник/ця стартапу і плануєш випустити на ринок мобільний застосунок для обміну світлинами котиків.

Яку методологію ти обереш для процесу розробки і чому? Відповідь текстово обґрунтуй.

Для цього стартапу я б використала ітеративну модель.

Можливо спочатку це буде виглядати як простенький додаток з елементарними функціями. Згодом можна розширити функціонал, тому що вимоги у користувачів будуть змінюватись (треба буде додати відео, live - опцію, мессенджер, редактор фото тощо).

Вважаю, що доцільно буде використати дану модель, тому що це буде тривалий проект і з часом необхідно буде вводити нові технології. При ітеративному методі буде зручно налагодити процес. Всі нововведення можна буде запровадити проводячи послідовні ітерації.