**Завдання 1.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Назва методології** | **Сильні сторони** | **Слабкі сторони** | **Для якої галузі є доцільною** |
| 1. | **Waterfall** | * Передбачуваність та прозорість системи, простота у вивчені; * Простота у контролі, тобто менеджер або замовник завжди розуміє, на якому етапі знаходиться розробка. * Легко планувати термін та бюджет розробки на початкових етапах; * Працюєш чітко дотримуючись вимогам документації, тому це не потребує якихось виключних знань та технічних навичок; | * Негнучкість методології; * Велика кількість документації; щоб внести якісь зміни в документи чи процеси, потрібно отримати багато узгоджень; Неможливість повернення на попередні етапи для внесення змін; * Не можливість приступити до роботи, поки не буде завершений попередній етап; * Тестування починається на достатньо пізніх етапах, і якщо на попередніх етапах була допущена помилка, то це буде важче і дорожче виправити; * Складність отримання зворотнього звя’зку від замовника на етапах розробки; Замовник бачить результат тільки в кінці, і не факт, що його цей результат влаштує; * Процес розробки займає багато часу до кінцевого результату, і продукт може втратити свою актуальність і стати не цікавим користувачам. | Проекти зі стабільними вимогами, де є чітке бачення результату. Наприклад, проекти, пов’язані з медициною, космічна або авіаційна галузь, банківська система, страхові компанії і т. д.  Або це може бути навпаки нескладні проекти, наприклад, інтернет-магазин, де не потрібно багато часу для розробки, тому можна зробити одним етапом. |
| 2. | **V-модель** | Удосконалена каскадна модель, тобто ті ж плюси, що і в каскадній моделі, але тут ще плюс, що планування тестування починається на ранніх етапах розробки, тому це зменшує витрати та час на усунення помилок | Так само як і в каскадній моделі неможливо повернутися на попередні етапи розробки і відсутність зворотнього зв’язку від замовника на етапах розробки | В проектах, де потрібна висока надійність, наприклад розробки систем безпеки для автомобілів, або система спостереження і т. д. |
| 3. | **Scrum** | * Гнучкість системи та адаптивність до змін; * Після кожної ітерації можна показати проміжний результат замовнику, отримати відгук та внести зміни за потреби; * Швидке вирішення задач командою; * Постійна комунікація команди, кожен може відслідкувати хто на якому етапі роботи і оперативно залучитись до своєї частини роботи; * Система не передбачає такої великої кількості документації, і в неї легше вносити зміни; * Можливість постійно покращувати продукт; | * Залученість клієнта-замовника до обговорення приводить до внесення постійних змін; * Відсутність чітких вимог та документації може привести до складнощів реалізації продукта; * Важко спрогнозувати кінцевий результат, терміни та бюджет проекту; * Постійні зустрічі та обговорення займає багато часу та сил, і залишає менше часу для виконання роботи, викликає роздратування у членів команди; * Ця модель компанії може бути некомфортною для людей-інтровертів і впливати на їх рішення чи залучатися до проекту; * Постійний контроль з боку Scrum-майстра * Найм Scrum-майстра та Product owner потребує додаткових витрат; | Може використовуватися в будь-яких сферах, і підходить для проектів з невизначеними чіткими вимогами, а також підходить для проектів, де залучено невелика команда розробників, так як Scrum передбачає невелику кількість людей в команді. Це може бути розробка інтернет-сайтів, мобільних додатків і т.д. |
| 5. | **Kanban** | * Також гнучка та адаптивна модель, має ті ж переваги, що і Scrum, але на відміну від Scrum не має такої кількості зустрічей, планування демонстрації проміжних результатів. * Менше контактів з замовником. * Підходить людям-інтровертам; * Прозора та чітка система постановки задач через візуалізацію; * Задачі розбиті на чіткі кроки * Зрозумілість задач допомагає легко ввести нового учасника команди в проект; * Обмеження кількості задач, які знаходять в процесі | * Відсутність чітких термінів виконання задач * Потребує від учасників високої самоорганізації, так як на відміну від Scrum не має того, хто здійснює контроль над виконанням задач, а це може впливати на ефективність і швидкість роботи | Технічне обслуговування, системне адміністрування, інженерія, промисловість; |
| 6. | **Спіральна модель** | * Розробка відбувається витками; * Поетапне покращення продукту, нарощення додаткового функціоналу на кожному новому витку; * Після кожного витка відбувається оцінка ризиків, тобто приймається рішення про доцільність продовження роботи над проектом, що мінімізує можливі додаткові фінансові витрати * Постійний аналіз ринку; * Орієнтація на потреби користувача, тобто чи буде продукт корисний користувачу, чи є сенс продовжувати далі покращення продукту; * Після кожного витка переоцінка термінів та витрат на розробку; * Поетапне покращення продукту, нарощення додаткового функціоналу; * Постійний аналіз ринку; | * Важка для розуміння структура * Нема чіткого розуміння кінцевого результату; * Важко визначити момент переходу на новий виток спіралі; | Може використовуватися в різних галузях, але здебільшого на проектах з високими ризиками і в дослідницьких проектах, або в проектах, в яких неможливо спрогнозувати кінцевий результат. |

**Завдання 2.**

Agile-маніфест був створений у 2001 році визначними фахівцями у галузі розробки ПЗ, які зібралися разом для обговорення поточного стану у галузі IT. Їх непокоїло те, що існуюча на той час модель розробки ПЗ не відповідала більше сучасним викликам. Консервативна модель Waterfall передбачала велику кількість документації та планування, розробка забирала багато часу, а відсутність можливості внесення змін на етапах розробки врешті-решт могли привести до результату, який би не відповідав інтересам замовника. Тому принципи Agile, як гнучної моделі, були прийняті для вирішення цих проблем. Нові принципи не відкидали важливість документації, чіткого планування, та узгодження умов контракту, але фокус уваги першочергово змістився на потреби замовника та кінцевого користувача. В центрі уваги гнучкої моделі став сам продукт і його користь та важливість для користувачів. Принципи Agile та методи, які були розроблені на її основі, пришвидшили процес розробки ПЗ, зробили можливим внесення змін на різних етапах, а можливість взаємодіяти з клієнтом мінімізували ризики створення продукту, який би не відповідав його очікуванням. Тому я вважаю, що принципи Agile, затверджені у маніфесті, зіграли позитивну роль у галузі розробки ПЗ та дали поштовх до створення багатьох сучасних методів управління проектами, таких як Scrum, Kanban, Lean і т.д. І хоча консервативні моделі управління проектами також лишаються актуальними на сьогоднішній день, але нові методи, основані на Agile, швидко завоювали свою популярність на ринку та практикуються у багатьох IT компаніях, що приносить позитивний результат для всіх учасників процесу, як розробників, так і клієнтів та користувачів.

**Завдання 3**

Як засновниця стартапу, що планує створити мобільний застосунок для обміну світлинами котиків, я би обрала ітераційну методологію управління, так як така модель не потребує багато документації, чітко встановленого бюджету, чіткого бачення кінцевого результату, та можливість вносити зміни до вимог на різних етапах розробки. Я би найняла невелику кількість розробників у команду, які відносно швидко зможуть випустити продукт в рамках однієї ітерації з якимось невеликим набором функціоналу. Після ітерації я зможу вивести продукт на ринок та вже отримувати прибуток від його експлуатації, а вже на наступних ітераціях можна вносити зміни у застосунок та напрацьовувати інші корисні функціонали, постійно покращуючи продукт. Для організації команди на проекті я би використала підхід Kanban з чіткою постановкою задач на період ітерації, бо на відміну від Scrum це не потребувало би додаткових фінансових витрат, таких як найм Scrum-майстра, Product owner.