***Порівняльна таблиця найбільш поширених методологій:***

| № | Назва методології | Сильні сторони | Слабкі сторони | Для якої галузі є доцільною |
| --- | --- | --- | --- | --- |
| 1 | Waterfall | * новий етап розробки не починається до тих пір, поки не завершиться попередній; * підходить для невеликих проєктів, де всі вимоги чітко окреслені; * на кожному етапі відбувається повне документування; * будь-які внесення змін можуть здійснюватися лише під час роботи над проєктом. | * дефекти можуть бути виправлені тільки на цьому етапі розробки; * не підходить для великих проєктів, де вимоги можуть змінюватися; * документування забирає багато часу; * будь-які зміни після завершення проєкту призводять до критичних збоїв. | * космічна галузь; * авіаційна галузь; * медицина |
| 2 | V-model | * фаза розробки та фаза тестування йдуть паралельно; * раннє знаходження та виправлення дефектів в межах однієї фази розробки; * чітко-сформульовані вимоги допомагають зберегти час процесу імплементації; * модель для чітко окресленого бюджету. | * вияв та виправлення помилок проводиться на ранніх стадіях реалізації системи; * ризики високі і непередбачувані; * не підходить для великих проєктів; * немає можливості для внесення змін після завершення фази; * у проєктах існують часові та фінансові обмеження. | * військові проєкти; * космічна галузь; * вбудовані системи (наприклад, ліфт) |
| 3 | Scrum | * раннє створення працюючого ПЗ * клієнт має доступ до розробки в будь-який час; * підходить для великих проєктів; * періоди роботи поділяються на спринти; * всі учасники є командними гравцями; * чітке розподілення ролей у команді (скрам-майстер, product owner, scrum-team); * команда швидко адаптується до змін, внесених замовником; * учасники команди працюють одночасно, наприклад: програміст створює код, в цей час тестувальник пише до нього тести; * для візуалізації завдань використовується дошка, вона очищається кожен раз при новій ітерації. | * не всі вимоги можуть бути відомі до початку розробки через це можуть виникнути проблеми із загальною архітектурою; * вимоги до проєкту можуть змінюватися в процесі розробки; * члени команди не мають права брати задачі на спринт, які не заплановані на пленінгу, а також змінювати пріоритети; * організація команди та ефективність виконання завдань залежить від менеджера/скрам-майстра | ІТ-сфера  мобільні застосунки  соціальні мережі |
| 4 | Kanban | * візуалізації роботи за допомогою дошки; * обмежує обсяг виконуваних робіт, тому команда фокусується на найважливішому; * нові елементи можна додавати за наявності ресурсів (а не лише після завершення всіх завдань) * не потрібно проводити щоденні наради * учасники команди можуть за потреби змінити напрямок робіт без жодного впливу на часову шкалу; * роботи здаються після завершення, що робить можливим постійне вдосконалення. | * пріоритети і оцінювання задач не обов'язкові; * без часових шкал в учасників команди може не виникнути відчуття нагальності чи потреби завершити завдання; * жодна особа не несе відповідальності за виконання командою цілей і дотримання рекомендацій; * без власника дошки можуть ставати надто складними й застарілими, що призводить до проблем із розробкою та плутанини. | ІТ-сфера  технічне обслуговування  системне адміністрування  маркетинг  продажі |

* **На твою думку, чому з’явився Agile-маніфест?**
* **Які проблеми він мав вирішити і чи це вдалося?**

На мою думку, Agile-маніфест з'явився у зв'язку зі стрімким розвитком технологічної галузі у 90-х роках, а отже необхідно було встановити чіткі принципи організації процесу розробки та суспільних взаємовідносин. Висока конкуренція та необхідність швидко видавати працюючий продукт поставили під сумнів каскадні моделі, які були актуальними довгі роки. За своєю сутністю підхід Agile є антиподом моделі Waterfall, що базується на таких принципах як наявність чіткої та докладної документації; беззаперечне слідування плану роботи; неможливістю повернення на попередні етапи з метою змін, а також виявлення та виправлення помилок тільки на етапі тестування.

Довгий час ці постулати були непорушними у сфері IT. Але коли бізнесу стали важливими такі аспекти, як швидкість одержання продукту та можливість його швидкої адаптації під поточні потреби ринку, “застаріла” методика Waterfall почала різко втрачати позиції.

Таким чином був створений, задокументований “Agile Manifesto” та опублікований у 2001 році.

Виконання рекомендацій, описаних у Agile Manifesto, змогли забезпечити проєкту можливість постійного тестування та трансформації. Використання принципів та ідей з “Agile Manifesto” заклало можливість постійних оновлень функціоналу та інших складових.

При використанні Agile вже на початкових етапах є продукт, що працює. І це дає такі переваги як тестування на ранніх стадіях; можливість оцінки доданого функціоналу "в дії"; дослідження досвіду користувача на всіх етапах; можливість швидкої презентації на ринку "сирої", але працюючої версії.

Методологія Agile враховує реалії робочих процесів, не ідеалізуючи їх. Замовники регулярно змінюють свої вимоги, підлаштовуючись під потреби ринку та потреби кінцевого користувача. Адаптивний підхід Agile передбачає зміни на будь-якій стадії розробки, що дозволяє отримати більш конкурентоспроможний продукт. Гнучка система також є гарним рішенням в умовах невідомості (скільки буде виділено фінансування, які фахівці працюватимуть, скільки знадобиться часу тощо).

При вдалому впровадженні цієї методології під час розробки це вирішило, ті питання та проблеми, які постали у 90-х роках.

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

Я оберу ітераційну модель, таку як, наприклад Scrum. тому що ця модель є гнучкою та передбачає на кожному етапі завершений продукт, яким споживач зможе користуватися. З часом до цього застосунки можна буде додати нові світлини чи нові функції, щоб споживач не втрачав інтересу до застосунку.