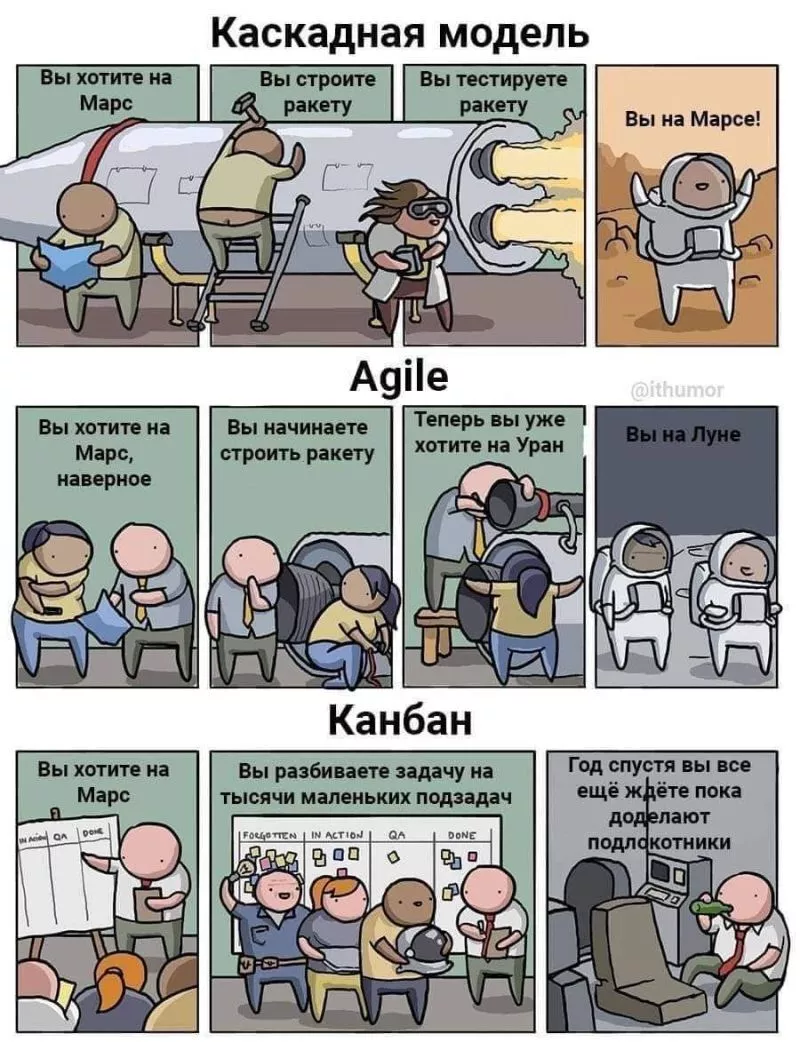
**Рівень 1.**

| **Технологія** | **Сильні сторони** | **Слабкі сторони** |
| --- | --- | --- |
| Waterfall | * дана модель стійка до оновлення кадрів (розробників можна замінити, на строки це не вплине); * ця модель дисциплінує, завдяки плану і чіткій послідовності етапів та суворому менеджменту; * модель гнучка на ранніх етапах (до етапу розробки можна вносити зміни в попередні етапи); * наперед зрозуміло, на якому етапі що буде відбуватися – простіше прогнозувати бюджети і набирати команду; * завдяки зручній звітності легко відслідкувати витрачений час, можливі ризики і ресурси в процесі роботи над проектом; * якість проекту займає першочергове місце, а витрачений час та бюджет відходять на другий план. | * велика кількість документації, яку треба постійно актуалізувати (робота над проектом перетворюється на бюрократію); * детальний план може створити хибне враження про продуктивність реалізації проекту; * користувача та замовника повністю ізолюють від процесу розробки (продукт не орієнтований на масового користувача); * всі вимоги повинні бути відомі одразу; * тестування відбуваться лише в самому кінці (на перших етапах розробки продукт може бути більш гнучким, але багато багів на етапі тестування можуть викликати проблеми в кінцевому продукті); * проект складно адаптувати під зміни середовища, план змінювати не можна. |
| Scrum | * мінімум зайвої бюрократії та непотрібної документації; * до співробітників прислухаються, вони мотивовані – щоденна видимість досягнень, прийняття рішень знаходиться в руках команди; * замовник отримає продукт, який сподобається аудиторії, адже він розроблений з урахуванням зворотнього зв’язку; * команда працює короткими етапами, на кожному з яких визначає цілі та шляхи їх досягнення – це пришвидшує процес роботи + працює над різними задачами проекту одночасно; * великі задачі поділяють на малі, тому внести корективи прямо в процесі роботи набагато простіше, ніж в каскадному підході; * скорочується час на пошук помилок та пояснення проблем; * мінімізація фінансових ризиків завдяки оперативній реакції на зміни та усунення помилок; * присутній відкритий обмін інформацією, що робить процес роботи максимально прозорим. | * в таку команду потрібні люди з досвідом – не у всіх є досвід роботи, на навчання співробітників треба витрачати ресурси; * нетабільна вартість продукту; * не підходить для великих проектів + є проекти, які вимагають виключно планового підходу в роботі; * успіх проекту багато в чому залежить від скрам-майстра, кваліфікації команди та їх відданості своїй справі; * проект вимагає регулярної комунікації із замовником, що часто уповільнює процес через неможливість отримання зворотнього зв’язку. |
| Канбан | * гнучкість планування - команда концентрується на одній конкретній задачі, керівник може змінювати пріоритетність роботи, не торкаючись робочого процесу; * дана методологія дозволяє відслідковувати робочий процес, оптимізувати його тривалість і прогнозувати час, який буде необхідний для вирішення мйбутніх задач; * даний метод мінімізує застої, оскільки допомагає швидко виявляти слабкі місця – це дозволяє скорочувати час роботи над завданням та підвищувати якість результату; * наглядність просування роботи – всі члени команди мають доступ до поставленим задачам і результативності просування, таким чином легше виявити проблеми та усунути їх. | * не підходить для довгострокового планування – метод розрахований на досягнення короткострокових цілей; * не підходить для великих команд; * відсутність структури – командам може бути важко працювати ефективно без певного набору практик та ролей – може призвести до плутанини та неефективності; * відсутність часових рамок – канбан не встановлює конкретну тривалість часу для роботи, це може утруднити ефективне планування і постановку пріоритетів; * відсутність розстановки пріоритетів – методологія не встановлює конкретні пріоритети для роботи, це може призвести до затримок та відсутності прогресу; * обмежена кількість ретроспектив спинту – утрудняяє вдосконалення робочих процесів. |
| Спіральна модель | * покращений аналіз ризиків; * якісна документація процесу розробки; * гнучкість – можливість внесення змін та додавання нової функціональності навіть на відносно пізніх етапах; * раннє створення робочих прототипів. | * може бут дорогою у використанні; * управління ризиками вимагає залучення висококласних фахівців; * успіх процесу у великій мірі залежить від стадії аналізу ризиків; * не підходить для маленьких проектів. |



**Рівень 2.**

*Чому з’явився Agile-маніфест? Які проблеми він мав вирішити і чи це вдалося?*

Agile-маніфест з’явився у 2001 р. та був сформований групою розробників ПЗ в м. Юта (США). Вони зібралися, щоб запропонувати нову методологію управління проектами з розробк ПЗ на противагу класичним бюрократичним підходам. Єдиної методології тоді не було вироблено, але з’явився Agile-маніфест – радикально новий підхід до управління проектами.

В другій половині ХХ ст. актуальною була Waterfall-технологія, антагоніст Agile. У той час дотримання каскадної технології було спричинено особливостями функціонування самого програмного забезпечення та процесом його розробки. Головна ідея полягала в тому, щоб прирівняти програмну інженерію до фізичної інженерії.

Методологія Waterall, як було задумано першочергово, мала враховувати зміни та перегляд проектних рішень. Передбачалася можливість повернутися до попереднього етапу та зкоректувати деякі рішення і очікування, і ці зміни могли змінити аспекти даного етапу. ле на практиці графіки і бюджети майже завжди змушували команди дотримуватися раніше прийнятих рішень.

На початку 1990-х рр., коли комп’ютери почали масово з’являтися на підприємствах, розробка ПЗ зіштовхнулася з кризою. Експерти підрахували, що час між підтвердженою потребою бізнесу та фактичним застосуванням ПЗ у виробництві складає близько трьох років. За цей час могли змінитися вимоги, системи і навіть сам бізнес. Тому багато проектів відмінялися на півдорозі, а багато завершених проектів, не відповідали актуальним потребам бізнесу. аме через це лідери в галузі розробки ПЗ почали проводити неформальні зустрічі і обговорювати способи більш простої розробки софту, без зайвих затрат на процес та документацію.

Лідери Agile шукали способи швидко створити працююче ПЗ та передати його в руки кінцевих користувачів. Так, користувачі швидше би отримували переваги нових версій, а команда розробки отримувала б швидкий зворотній зв’язок про продукт.

Зараз є різні думки щодо ефективності методології Agile і на її ефективне впровадження впливає багато факторів. Серед причин провалу гнучкої методології можна назвати некомпетентність учасників (команда має складатися з мотивованих професіоналів), нерозуміння їх ролей, незрозумілі вимоги, внесення змін у вже працюючі/тестовані версії.

В свою чергу, розробка короткими ітераціями та швидкий фідбек від замовника дійсно дозволяє адаптуватися до змінним вимогам. Agile дозволяє зробити роботу більш передбачуваною, дозволяє експерименти, дає можливість не робити зайвої роботи (особливо щодо бюрократичної тяганини з документаціями, погодженнями і т.д.). Agile – активна робота з невідомими, можливість пропрацьовувати ризики через активний зворотній зв’язок. Адже одна з головних проблем – не встигнути, випустити продукт, який вже нікому не потрібен. З поширенням гнучких підходів в ІТ-проектах почастішали спроби скористатися їхніми перевагами в інших сферах, де також можливі невизначеність та часті зміни. Зараз Agile впроваджується в продажах, маркетингу, рекрутингу та управлінні талантами. Ініціативи в цих напрямках об’єднують навколо себе спеціалістів різних профілів, утворюючи так звані багатофункціональні команди.

**Рівень 3.**

*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.*

Головна мета стартапу, на мою думку, якомога швидше вийти на ринок з готовим до використання програмним забезпеченням і стати на шлях повернення інвестицій. Каскадні технології не дуже підходять в цій ситуації, адже являють собою чітку структуру і вимагають великої затратності часу, що не дуже підходить для молодого та гнучкого стартапу.

Тому, в даній ситуації, більш підходящими є гнучкі моделі. Адже якщо ми будемо працювати за гнучкою моделлю, наш проект та вимоги до нього можуть змінюватися протягом кожної ітерації. Так, наприклад, Scrum і Kanban мають відрізняються відносною простотою впровадження, застосування та розуміння з боку команди. Також ці методології підходять по критеріям розміру команди (в стартапі вона не буде дуже велика) та фокусом на якість.

Виходячи з особливостей продукту, який орієнтований перш за все на задоволення потреб споживачів, та особливостей стартапу, можна обрати гібридну модель на основі Scrum i Kanban.

Команди розробників можуть бути згруповані за напрямками роботи. Окрім того, можна ввести посаду Product Owner – власника продукту, який додатково буде виконувати функцію контролю виконання правил (Scrum Master). Весь проект можна розподілити на місячні ітерації, для різних напрямів, та проводити опитування фокус-групи в кінці кожного спринта.

Буде створено певний перелік побажань проекту, в який буде занесено майбутні задачі та задачі, які виникатимуть за результатами опитування фокус-групи. Згодом ці задачі будуть пріоритизовані, при спільному обговоренні. З даного переліку задач (журналу задач) будуть обиратися задачі для наступного спринта і розставлятися за пріоритетністю в колонці “To do”, в програмі Trello, наприклад.

Також зі Scrum можна взяти 15-хвилинні зустрічі команди на початку робочого дня, та аналіз виконаної роботи в кінці спринту. Цю ідею я запозичила в Інтернет-джерелах, але вона здалая мені відносно реальною і найбільш обгрунтованою зі всіх поглядів.

