**Beet Seed  — відпрацюй навички на базовому рівні.**

**Склади порівняльну таблицю найбільш поширених методологій:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Назва методології** | **Сильні сторони** | **Слабкі сторони** | **Для якої галузі є доцільною** |
| **1** | **Waterfall** |  |  |  |
| **2..** |  |  |  |  |

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

***б. Відповіді запиши в той самий файл Google Docs. Додай посилання на нього в LMS.***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Назва методології** | **Сильні сторони** | **Слабкі сторони** | **Для якої галузі є доцільною** |
| 1 | «Waterfall Model» (каскадна модель або «водоспад») | * модель добре відома споживачам; * вона впорядкування справляється зі складнощами і добре спрацьовує для тих проєктів, які досить зрозумілі, але все ж важко розв'язні; * вона досить доступна для розуміння вона проста і зручна в застосуванні, оскільки процес розроблення виконується поетапно; * її структурою може керуватися навіть слабо підготовлений у технічному плані або недосвідчений персонал; * вона вирізняється стабільністю вимог; * вона являє собою шаблон, у який можна помістити методи для виконання аналізу, проектування, кодування, тестування та забезпечення; * за правильного використання моделі дефекти можна виявити на більш ранніх етапах, коли їх усунення ще не вимагає відносно великих витрат; * вона визначає процедури з контролю за якістю. Кожні отримані дані піддаються огляду. Така процедура використовується командою розробників для визначення якості системи; * стадії моделі досить добре визначені та зрозумілі | * в основі моделі лежить послідовна лінійна структура, внаслідок чого кожна спроба повернутися на одну або дві фази назад, щоб виправити будь-яку проблему або недолік, призведе до значного збільшення витрат і збою в графіку; * користувачі не можуть переконатися в якості розробленого продукту до закінчення всього процесу розробки. Вони не мають можливості оцінити якість, якщо не можна побачити готовий продукт розробки; * у користувача немає можливості поступово звикнути до системи. Процес навчання відбувається наприкінці життєвого циклу, коли ПЗ уже запущено в експлуатацію; * кожна фаза є передумовою для виконання наступних дій, що перетворює такий метод на ризикований вибір для систем, які не мають аналогів, оскільки він не піддається гнучкому моделюванню; * для кожної фази створюються результативні дані, які після її завершення вважаються замороженими. Це означає, що вони не повинні змінюватися на наступних етапах життєвого циклу продукту. Якщо елемент результативних даних якого-небудь етапу змінюється (що трапляється досить часто), на проєкт негативно вплине зміна графіка, оскільки ні модель, ні план не були розраховані на внесення і дозвіл зміни на пізніших етапах життєвого циклу; * усі вимоги мають бути відомі на початку життєвого циклу. Модель не розрахована на динамічні зміни у вимогах протягом усього життєвого циклу, оскільки одержувані дані "заморожуються". виникає необхідність у жорсткому управлінні та контролі, оскільки в моделі не передбачена можливість модифікації вимог; * модель заснована на документації, а отже, кількість документів може бути надмірною; весь програмний продукт розробляється за один раз. Немає можливості розбити систему на частини. | ПЗ для медичного обладнання (бо для кожного нового обладнання потрібне нове більш досконале ПЗ) |
| 2 | V-Model | - у моделі передбачено атестацію та верифікацію всіх зовнішніх і внутрішніх отриманих даних, а не тільки самого програмного продукту;  - у V-подібній моделі визначення вимог виконується перед розробкою проєкту системи, а проєктування ПЗ - перед розробкою компонентів;  - модель визначає продукти, які мають бути отримані в результаті процесу розроблення, причому кожні отримані дані мають піддаватися тестуванню;  - завдяки моделі менеджери проєкту можуть відстежувати перебіг процесу розроблення, оскільки в цьому разі цілком можливо скористатися часовою шкалою, а завершення кожної фази є контрольною точкою;  - модель проста в застосуванні. | - з її допомогою непросто впоратися з паралельними подіями;  - у моделі не передбачено внесення вимоги динамічних змін на різних етапах життєвого циклу;  - тестування вимог у життєвому циклі відбувається занадто пізно, внаслідок чого неможливо внести зміни, не вплинувши при цьому на графік виконання проєкту;  - у модель не входять дії, спрямовані на аналіз ризиків. | ПЗ в оборонній галузі Німеччини (через жорсткий контроль всіх вимог і етапів розробки) |
| 3 | Ітеративна модель (iterative model) | * раннє створення працюючого ПЗ; * гнучкість – готовність до зміни вимог на будь-якому етапі розробки; * кожна ітерація – маленький етап, для якого тестування та аналіз ризиків забезпечити простіше, ніж для всього життєвого циклу продукту. | * кожна фаза – самостійна, окремі ітерації не накладаються одна на одну; * можуть виникнути проблеми з реалізацією загальної архітектури системи, оскільки не всі вимоги відомі до початку проектування. | Мобільні додатки (в цьому випадку є можливість динамічно розвивати проект, постійно поліпшувати функціонал та адаптувати його до потреб користувачів |
| 4 | Спіральна модель | * спіральна модель дозволяє користувачам "побачити" систему на ранніх етапах; * ця модель дозволяє користувачам активно брати участь під час планування, аналізу ризиків, розроблення, а також під час виконання оціночних дій; * у моделі передбачено можливість гнучкого проєктування реалізовано переваги інкрементної моделі, а саме випуск інкрементів, тут не ставиться мета виконати неможливе - довести конструкцію до досконалості; * зворотний зв'язок у напрямку від користувачів до розробників виконується з високою частотою і на ранніх етапах моделі, що забезпечує створення потрібного продукту високої якості; * підвищується продуктивність завдяки використанню придатних для повторного використання властивостей; * підвищується ймовірність передбачуваної поведінки системи за допомогою уточнення поставлених цілей; * при використанні спіральної моделі не потрібно розподіляти заздалегідь усі необхідні для виконання проєкту фінансові ресурси; | * якщо проєкт має низький ступінь ризику або невеликі розміри, модель може виявитися дорогою. Оцінка ризиків після проходження кожної спіралі пов'язана з великими витратами; * модель має ускладнену структуру, тому може бути ускладнене її застосування розробниками, менеджерами і замовниками; * серйозна потреба у високопрофесійних знаннях для оцінки ризиків; * спіраль може тривати до нескінченності, оскільки кожна відповідна реакція замовника на створену версію може породжувати новий цикл, що віддаляє закінчення роботи над проєктом (ухвалення спільного рішення про припинення процесу розробки); * велика кількість проміжних стадій може призвести до необхідності в обробці внутрішньої додаткової та зовнішньої документації; * використання моделі може виявитися дорогим і навіть неприпустимим за коштами, оскільки час, витрачений на планування, повторне визначення цілей, виконання аналізу ризиків і прототипування, може бути надмірним; * при виконанні дій на етапі поза процесом розроблення виникає необхідність у перепризначенні розробників; * можуть виникнути труднощі при визначенні цілей і стадій, що вказують на готовність продовжувати процес розроблення на наступній ітерації. | Розробка системи документообігу для банківської установи (кожен наступний крок вимагає більшого аналізу для оцінки наслідків, ніж програмування) |
| 5 | «Incremental Model» (инкрементная модель) | * Програмне забезпечення буде генеруватися швидко протягом життєвого циклу програмного забезпечення. * На всіх етапах розробки можуть бути внесені зміни * Ця модель дешевша порівняно з іншими * Клієнт може відповісти на кожне завдання * Помилки легко ідентифікувати | Це вимагає хорошого планування проєктування  Проблеми можуть виникнути через архітектуру системи, тому не всі вимоги зібрані заздалегідь за весь життєвий цикл програмного забезпечення.  Кожна фаза ітерації жорстка і не перекриває одна одну  Усунення проблеми в одній одиниці вимагає виправлення у всіх одиницях і займає багато часу | Електронні бібліотеки (повинно постійне розширення серверів та функціоналу для максимального вдосконалення використання) |

**Beet Sprout — 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?**
* **Які проблеми він мав вирішити і чи це вдалося?**

11-13 лютого 2001 року на гірськолижному курорті The Lodge at Snowbird у горах Восатч у штаті Юта сімнадцять осіб зустрілися, щоб поговорити, покататися на лижах, розслабитися і спробувати знайти спільну мову - і, звісно ж, поїсти. Так з'явився Agile-маніфест "Розробка програмного забезпечення". Зібралися представники компаній Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming та інших, що відчували потребу у створенні альтернативи великоваговим процесам розроблення програмного забезпечення, що базуються на документації.

Наразі важко було знайти більше зібрання організаційних анархістів, ніж те, що вийшло на цій зустрічі, було символічним - Маніфест гнучкого розроблення програмного забезпечення, підписаний усіма учасниками.

Маніфест звучить так:

*«Ми постійно відкриваємо для себе досконаліші методи розробки програмного забезпечення, займаючись розробкою безпосередньо та допомагаючи у цьому іншим.  
Завдяки цій роботі ми змогли зрозуміти, що: Люди та співпраця важливіші за процеси та інструменти. Працюючий продукт важливіший за вичерпну документацію.  
Співпраця із замовником важливіша за обговорення умов контракту  
Готовність до змін важливіша за дотримання плану. Тобто, хоча, цінності, що справа важливі, ми все ж цінуємо більше те, що зліва».*

Назвавши себе "Аджайл Альянс", ця група незалежних мислителів про розробку програмного забезпечення, а подекуди й конкурентів один одного, погодилася з Маніфестом гнучкого розроблення програмного забезпечення.

Сам підхід передбачає більш гнучку розробку продукту вмежах кожного циклу (ітерації) з більш тісною взаємодією з замовником, що дозволить максимально задовольнити його потреби.

З часом цей підхід був доопрацьований і на його основі були розроблені такі методи управління розробкою проектів як Scrum, Kanban та інші. Таким чином, можна констатувати, що основні завдання, які стояли перед творцями маніфесту, коли вони його тільки створювали, були вирішені, але через деякий час виникла потреба вдосконалення існуючого метода шляхом створення на його основі більш досконалих методик.

**:лиственное_дерево:Mighty Beet — 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.** |

В описаному випадку потрібно вибрати модель розробки, яка б давала змогу тісно взаємодіяти з користувачами та змінювати продукт у процесі його використання. Найбільш доцільною для цього буде ітеративна модель.

Ітеративна модель підійде для тих, хто:

* розробляє MVP-версію (minimum viable product, MVP) - продукт, що володіє мінімальними, але достатніми для задоволення перших споживачів функціями) продукту для оцінювання концепції та пошуку product/market fit;
* працює над невеликими стартапами й поки не знає, яким має бути продукт;
* хоче створити продукт, що задовольнятиме конкретні потреби;
* зацікавлений у постійному розвитку свого проєкту.

З іншого боку, необхідно враховувати наступні нюанси такого підходу:

* не кожен клієнт погодиться на такий підхід, більшість віддає перевагу класичним методам розробки;
* потребує багато часу, тому можна не вкластися у визначений бюджет і дедлайн;
* потенційні проблеми з архітектурою, оскільки вимоги до продукту постійно змінюються. Крім того, щоб надати їй адекватного вигляду, можуть знадобитися додаткові ресурси.

При розробці додатку команда повинна враховувати наступні особливості:

1. Discovery Stage — це попередній етап, на якому виявляється цільова аудиторія, її потреби та проблеми. Основна мета — підтвердити або спростувати припущення щодо продукту та зрозуміти, чи вирішує він проблему користувача. На початковому етапі треба скласти приблизні Customer Journey Map (CJM) і Proto Persona. Для цього необхідно досконало знати аудиторію, на яку орієнтований стартап: чим живуть потенційні користувачі, з якими проблемами стикаються. Щоб скласти «персону» користувача, бажано провести невелике інтерв’ю з потенційною аудиторією. Через живу комунікацію можна зрозуміти, що найбільше турбує (pains) аудиторію і як можливо допомогти це виправити. Дослівно Customer Journey Map — це мапа подорожі користувача. Вона показує точки дотику людини з продуктом, тобто ті етапи, які долає користувач для досягнення своєї мети. У нашому випадку CJM постійно оновлювалася впродовж проекту. Це дозволить підлаштуватися під користувачів і розв’язати нові проблеми в кожній ітерації. На базі виявлених під час інтерв’ю потреб і проблем користувачів можна буде скласти гіпотези та протестували їх. Лише після цього можливо буде побачити реальну картину того, хто наш користувач і які в нього потреби.
2. Після тестування MVP з фокус-групою визначаються подальші вимоги до продукту. Код у такому разі не має бути на 100% ідеальним. Утім він має виконувати своє цільове призначення та давати змогу провести тестування. Про деталі дизайну можна подбати пізніше. Наприклад, спочатку можна мати лише нарис функції, щоб перевірити, як нею користується аудиторія. Скажімо, прототип кнопки, що надсилає фото котика іншому користувачеві в чаті. Коли людина натискає на кнопку, вона бачить повідомлення про те, що функція перебуває в розробці й буде доступна найближчим часом.Отже, можна таким чином дізнатися, наскільки вона є затребувана серед учасників фокус-групи, не розробляючи функціонал повністю. Такий підхід дає можливість задати вектор розвитку продукту та зменшити витрати через попередню валідацію ідей та функцій.
3. Із самого початку важливо знайти так званих early adopters — групу потенційних користувачів, які тестуватимуть застосунок і надаватимуть зворотний зв’язок. Звісно, не всі з обраних учасників точно потраплять у цільову аудиторію. Втім, що раніше почнеться тестування гіпотези з реальними користувачами, то швидше можливо зрозуміти, хто є цільовою аудиторією, для якою розробляється стартап.
4. Проект розвиватиметься повільно, якщо команда не має чіткого бачення продукту та не знає, як рухатися далі. Тому варто долучати всіх членів команди до проведення Discovery Stage, визначення цільової аудиторії та основної проблеми, яку вирішує продукт. На основі цих даних сформується загальне бачення майбутнього проекту та його функціоналу. Як уже зауважували, в ітеративній моделі вимоги до продукту постійно змінюються. Але можна вносити такі зміни та візуалізувати прогрес завдяки roadmap.

Дорожня мапа дасть змогу команді знати:

* у якому напрямку рухається проєкт;
* на якому етапі перебуває;
* чого треба досягти найближчим часом;
* пріоритетність завдань.

При такому плануванні кожен учасник має перед собою довгострокову стратегію та може сам вносити ідеї щодо того, як швидше досягти результату. Це особливо ефективно для Agile-команди, де на кожному етапі розробки продукту залучені спеціалісти з різних сфер.

Отже, ітеративна модель — це найбільш ефективний підхід в даному випадку, особливо якщо є намір постійно вдосконалювати та розвивати проект, поліпшувати функціонал та адаптувати його до потреб користувачів.