Залікова робота

Залікова робота з предмету “Управління проектами”

студента 2 курсу групи МП-21

Захарова Дмитра

Варіант 6.

Завдання 1. Модель Agile основні риси, переваги та недоліки, особливості застосування.

Відповідь. *Agile* — це не одна окрема методологія, а скоріше ціла група методолгій керування, головна особливість яких — гнучкість та адаптивність. З’явилася після того, як *Waterfall* була недостатньо ефективною у рамках складних проектів. З’явилася у 2001 році, коли кілька IT представників випустили *Agile*-маніфест. За ідеологією є повною протилежністю *Waterfall*.

Переваги.

- Гнучкість та свобода — не потрібно чітко позначати етапи та наголошувати на вимогах, тому виконавці можуть експрементувати та вносити зміни поступово.
- Знижений ризик — ця методолгія має на увазі постійне отримання зворотнього зв’язку від клієнтів та членів команди, що значно скорочує ризик провалу проекту, оскільки необхідні ресурси залучені до процесу.

Недоліки.

- Реакція на зміни відбувається тоді, коли виникають проблеми.
- Відсутність чіткого плану, що ускладнює управління проектом та ресурсами.
- Доводиться постійно балансувати ресурсами, переводити ресурси з одного завдання на інші.
- Складність взаємодії — оскільки чіткого плану немає, всім зацікавленим сторонам доведеться працювати у дуже тісній співпраці, щоб усі розуміли статус проекту

Для яких проектів можна застосовувати.

- Коли немає впевненості, яким буде кінцевий результат, але є загальне розуміння ідеї.
- Коли взаємодія та комунікація — сильні сторони, а планування — ні)
- Коли потрібно проект адаптувати до швидких змін.

Приклади *Agile* методологій: Scrum, Kanban.

Завдання 2. Стискання розкладу.

Стиснення розкладу — це скорочення тривалості розробки проекту без зміни змісту самого проекту, часових обмежень, різних статусних дат тощо. Методи стиснення містять у собі:

- **Стистення.** Аналізуються компроміси між вартістю й розкладом для визначення, яким способом максимально стиснути строки при мінімальних витратах. **Приклади** — схвалення понаднормової роботи, використання додаткових ресурсів або плату за прискорення поставки для операцій. Ефективно тільки якщо додаткові ресурси дійсно здатні скоротити тривалість виконання завдання. Не завжди створює життєздатну альтернативу й може призвести до збільшення ризиків. Звісно збільшує вартість та бюджет проекту (згадуємо золоту формулу $\mathrm{Cost} + \mathrm{Scope} + \mathrm{Time} = \mathrm{Quality}$).
- **Швидкий прохід.** При цьому методі стиснення розкладу фази або операції, ті завдання, що мали б виконуватись послідовно, тепер виконуються паралельно. Звісно ця методика може бути використана лише якщо завдання можна розпаралелити (тобто якщо одна задача не блочить іншу). Також ця методика сильно підвищує ризики та вирогідність багів та помилок у різних частинах проекту.

Насправді менеджмент під час стиснення розкладу — це дуже індивідуальна річ для кожного PM’у, і немає універсальних методик або інструментів, які здатні розв’язати будь-яку конкретну проблему. Зазвичай, варто дотримуватися наступних принципів в таких критичних ситуаціях:

- Розбити всі завдання на менші частини, або точно розуміти об’єм і послідовність виконання.
- Ставити пріоритет на критичних тасках: потрібно розуміти, на чому робити акцент, а де можна “схалявити” і витратити менше часу, аби далі вже після

дедлайну відредагувати.

- Розуміти довжину кожного з таску. Якщо таск короткий, його наврядше можна якось розбити або пришвидшити, а ось великий можна розбити на малі і знайти якусь альтернативу. Взагалі, останні 3 пункти можна об’єднати в один: мати дуже конкретне бачення усього проекту, **особливо** під час стиснення розкладу.
- Збільшити об’єм комунації з командою — потрібно постійно розуміти темп розробки, статус та дуже швидко адаптуватися в залежності від конкретної ситуації.
- Інвестувати в технології — використовувати інструменти для менеджменту, аби краще розуміти стан проекту. Сюди ж можна віднести покупку та використання якихось *External APIs* — якщо немає часу розробляти систему власноруч, іноді значно зручніше використати та заплатити за вже розроблену *API*.
- Якщо зміна або переогляд графіку ніяк не допомогає покращити час виконання — варто долучити більше розробників, проте варто враховувати тривалість *onboarding* учасника, бо дуже швидко адаптуватись людині на новому проекті дуже важко.
- Краще як можна раніше розуміти, коли потрібно стискати графік, аби адаптувати зміни. А краще просто планувати так, щоб цього не доводилось робити 🙂

Завдання 3. Мотивація команди.

Мотивація — це процес, керуючий поведінкою людини, що задає спрямованість діяльності, активність, стійкість.

Мотивація є основною творчої роботи, тому дуже важливо підтримувати її на високому рівні.

Загалом виділяють дві **групи мотивів** діяльності людини:

- *Зовнішні* — прага до матеріальних ресурсів
- *Внутрішня* — задоволення від творчого процесу, естетичне задоволення

Звісно для значного покращення якості продукту важливо задовольняти внутрішню мотивацію людини, проте на жаль, в більшості випадків компанії забезпечують лише перше, і то не завжди.

Зазвичай роль фасілітатора (фасилітація — підвищення активності людини внаслідок наявності зовнішнього середовища, а фасілітатор — той, хто фасиліцює) бере на себе PM. Цей ефект полягає в тому, що прості завдання краще виконуються при наявності соціального оточення та його реакції. Проте, для складних проектів ефект може бути протилежним.

Ціль фасиліції — знайти правильний підхід, що дозволить групі працювати більш результативно і конструктивно. Насправді головний секрет фасилітації у тому, що найбільші ефективні рішення ті, які люди можуть виробити самостійно.

Фасилітатор приймає на себе наступні **ролі**:

- *Мотиватор* — знайти учасникам особистий сенс у дискусіях.
- *Модератор* — логічно вести команду по процесу розробки.
- *Медіатор* — будує максимально відкриту середу для обміну ідеями та знаходження консенсусу між ними.
- *Аналітик* — аналізувати ідеї учасників та задавати питання, аби корегувати “курсом” дискусії.
- *Ментор* — підтримувати учасників за їх результати та старання.

Причому головне у цьому всьому — фасилітатор є нейтральною стороною, тобто він слугує лише “штурманом” на проекті, що допомогає його вести в правильний бік, до власного рішення учасників, але ніяким чином не навчає цю групу або команду.

До **методів фасилітації** можна віднести:

- Робота в міні-командах, що дозволяє синхронізувати думки усіх учасників, легше прийти до консенсусу за більш короткий проміжок часу.
- Візуаліація процесу на дошці, діаграмах тощо.
- Мозковий штурм і подальше сумування згенерованих ідей.
- Вихід за рамки — часткова відмова від бюрократії, пошук методів для покращення бізнес-процесів всередині компанії.
- Поляризація поглядів — використання “крайностей”, можливість “спустити пар”

Головні плюси:

- Заохочення до демократичного способу прийняття рішень.
- Можливість підняти групову динаміку.

- Зниження стресу в інтровертних учасників.
- Економія часу і ресурсів компанії.
- Натхнення учасників на творчі ідеї.
- Можливість знайти найкраще рішення у вигляді консенсусу багатьох ідей.
- Створення безпечного психологічного простору в колективі.
- Постійний обмін досвідом між учасниками.

Звичайно, це все є плюсами лише за умілого використання принципів фасилітації PM’ом. Недоліков у цієї методології сильно немає — можна хіба що віднести ефект соціальної лені.

Обмеження (коли не варто використовувати фасилітацію):

- Є острий дефіцит часу.
- Команда біля дедлайну.
- Якщо у команди зовсім не розвинені навики комунікації.
- Якщо у учасників немає стійкої довіри до фасилітатора.
- Немає сенсу фасилітувати процеси, у яких не буде підтримки.

Завдання 4. Онлайн іструменти управління проектами: *Wireframing Tools*.

Іноді для розробки проекту важливо швидко “накинути” прототип продукту. Для цього потрібні так звані W*ireframing Tools*, тобто інструменти, що дозволяють створити прототип-лейаут веб сторінки або додатку без фокусу на окремі деталі.

Перерахуємо декілька таких тулзів:

- **Balsamiq** — сервіс для створення скеточваних макетів сайтів, вайфреймів, додатків та інших інтерфейсів. Дозволяє зробити скетч додатку або сайту дуже швидко, не відволікаючись на зайві деталі або непотрібні елементи. Робиться це за допомогою графічного редактора, де є робоча поверхня та панель інструментів, з яких перетягуються елементи макета. Елементів є доволі багато: кнопки, контейнери, шари, іконки, усілякі елементи, притаманні конкретним мобільним операційним системам тощо. Макети можна створювати спільно, учасники можуть як редагувати, так і коментувати і робити нотатки. З особливостей можна відмітити експорт

макетів у *PNG/PDF*, перетягування drag-and-drop, інтеграція з *Google Disk/Jira*.

- **Sketch** — сервіс виключно для Mac користувачів, має дуже простий і інтуїтивний інтерфейс на відміну від важких *Affinity Designer* або *Adobe Illustrator*. Має велике community, що дозволяє скачувати вже готові шаблони, на основі яких можна будувати мокап сайту. Є підтримка роботи в команді, можливість коментувати. Проте, немає вбудованих готових UI елементів — є лише базові геометричні фігури та текст (по суті, як в *Figma*)
- **Adobe XD** — в принципі йому притаманні усі плюси Sketch, але він доступий ще для Windows користувачів. З особливостей можна виділити те, що Sketch має вбудовану функцію конвертації структури у *HTML+CSS*, що може зберегти час.
- **Figma** — мабуть найбільш популярна програма з усіх, дуже зручна, мінімалістична, легка у використанні, часто використовується не лише для *wireframing*'у, а і для розробки повноцінного дизайну. Рекомендується використовувати для команд з малим бюджетом, бо має дуже широкий спектр функціоналу навіть для безкоштовної версії. Є усі потрібні функції колабораційного використання — можна редагувати разом, коментувати, через браузерний сайт дивитися з перспективи (камери) інших учасників. Має інтеграцію з дуже великою кількістю інших додатків (як *Notion*, наприклад).
- **UXPin** — той самий набір функціоналу, що і у додатків до цього. З особливостей можна виділити простоту, доволі широкий спектр функціоналу для саме демонстрацій, можливість редагувати елементів за допомогою *HTML* та *CSS*.