|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Homework\_2 Olga Homework\_2 Olga\_Laikova QA\_117\_Lykashеvych | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
| 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. | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
| Відповідь: | |  |  |  |  |  |  |  |  |  |
|  | | | | | | | | | | |
| № | Назва методології | Сильні сторони | | | Слабкі сторони | | | Для якої галузі є доцільною | | |
| 1 | Waterfall - послідовне проходження етапів розробки, перехід на новий етап можливий тільки після закінчення попереднього | Повне документування кожного етапа, дозволяє строго зафіксувати бюджет і строки розробки, прозорість процесів для замовника, якщо в команду прийде новий співробітник, йому буде легко розібратися, оскільки в наявності документація. | | | Оскільки важко передбачити всі проблеми на проекті з самого початку, і тестувальник допускається до продукту тільки на етапі тестування, тому знайдена помилка коштує дорого, оскільки потрібно заново проходити всі етапи, не передбачено внесення змін протягом виробництва | | | У невеликих проєктах, авіабудування, військова космічна галузь, медицина, фінансовий сектор, тому що цим сферам потрібні чітко вибудувані процеси і терміни для державних замовлень, на початку є чітке розуміння того, яким повинен бути кінцевий продукт | | |
| 2 | V-model - послідовне проходження етапів як в waterfall, на кожному етапі відбувається контроль поточного процесу | Повне документування кожного етапа, дозволяє строго зафіксувати бюджет і строки розробки, прозорість процесів для замовника, тестування відбувається на ранніх етапах, що підвищує якість тестування | | | Оскільки тестування відбувається після кожного етапу, це подовжує час виробництва продукту, , не передбачено внесення змін протягом виробництва | | | У невеликих та середніх проєктах, авіабудування, військова космічна галузь, медицина, фінансовий сектор, тому що цим сферам потрібні чітко вибудувані процеси і терміни для державних замовлень, на початку є чітке розуміння того, яким повинен бути кінцевий продукт | | |
| 3 | Ітераційна - проект складається з ітерацій, поступове наближення до образу готового продукту, на кожній ітерації повторюється модель waterfall. | Раннє створення працюючого ПЗ, готовність до внесення змін на будь-якому етапі розробки, для кожного маленького етапу легше забезпечити тестування та аналіз ризиків. | | | Можуть виникнути проблеми з реалізацією загальної архітектури системи, оскільки не всі вимоги відомі до початку проєктування, етапи не перетинаються. | | | Підходить для великих проектів з невизначеними вимогами та для іноваційних продуктів, які ще потребують перевірки. | | |
| 4 | Інкрементальна - розробка проекту закінченими частинами, які інтегруються в систему. Випуск на першому великому етапі продукту в базовій функціональності, а потім уже послідовне додавання нових функцій | Раннє створення працюючого ПЗ, готовність до внесення змін на будь-якому етапі розробки, для кожного маленького етапу легше забезпечити тестування та аналіз ризиків. | | | Оскільки створення деяких модулів буде завершено значно раніше за інші, виникає необхідність у чітко визначених інтерфейсах, не передбачені ітерації в рамках кожного інкременту | | | Для проектів де більшість вимог можна сформулювати заздалегідь, але їхня поява очікується через певний період часу, а також якщо ринкове вікно надто "вузьке" і існує потреба швидко поставити на ринок продукт, що має функціональні базові властивості; | | |
| 5 | Спіральна - розробка ПЗ має вигляд серії послідовних ітерацій, на кожному витку створюється чергова версія, після кожної ітерації проводиться оновлена оцінка ризиків щодо продовження розробки, на етапі аналіза рисків створюються прототипи, макети. | Раннє створення працюючого ПЗ, готовність до внесення змін на будь-якому етапі розробки, часті релізи нових версій ПЗ | | | Є ризик невиконання бюджету або не вкластися в терміни, дорога модель, більше документації | | | Дуже дорога модель, тому доцільно використовувати для середніх або великих проектів, які мають великі ризики, наприклад дослідницькі, або в великих проектах де оцінка кожного наступного кроку займає більше часу ніж програмування (система документообігу для банку) | | |
| 6 | Scrum - ітерації називаються спрінтами (1-4 тижні); 3 ролі: власник продукту, скрам-майстер, команда 8-9 чоловік; беклог спринта; 3 види зустрічей: перед спринтом для визначення задач, під час спринта- daіly meetings (15хв) - Що зробив?, Що роблю зараз? Що блокує?; в кінці спринта - демонстрація продукта та мітинг ретроспетиви; дуже важливо встигнути виконати задачі до кінця спринта, щоб були чіткі видимі результати. | Раннє створення працюючого ПЗ з основними функціями; повна взаємодія учасників команди, розподіл навантаження всередині команди; на кожний спринт визначаються нові цілі в залежності від приорітету та методи вирішення задач, це прискорює роботу; у замовника є повний контроль над процесом, може на кожному спринті додати нові задачі, можна постійно робити оновлення. | | | Вимагає регулярної комунікації із замовником, що часом гальмує процес через неможливість отримання зворотного зв'язку; не передбачає наявність фіксованого бюджету і фіксованого технічного завдання, що ускладнює юридичне оформлення такого роду домовленостей; практично немає документації. | | | більше підходить для проєктів, де потрібен чіткий графік виконання задач, які вимагають швидкого реагування на зміни ринку, щоб випустити продукт раніше інших, та в яких на початку може бути тільки ідея, замовник може не усвідомлювати як має виглядати кінцевий результат | | |
| 7 | Kanban - конвеєр задач, 3 правила: візуалізація процеса за допомогою канбан дошки, обмеження на кількість завдань на кожному етапі; постійне вимірювання продуктивності команди та покращення; завдання можна додавати хоч щодня | Раннє створення працюючого ПЗ з основними функціями; повна взаємодія учасників команди, розподіл навантаження всередині команди; кожен член команди може відслідковувати етапи проходження кожної задачі; зручно виявляти недосконалості під час процесу і корегувати завдання чи кількість виконавців. | | | не підходить для великих команд, оскільки в цьому випадку складніше контролювати виконання задач, не підходить для довгострокового використання; практично немає документації. | | | зручно використовувати для проектів на етапі технічної підтримки, коли завдання можуть з'являтись хоч кожного дня і потребують вирішення незалежно одна від одної | | |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| 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? | | | | | | | | | | |
| Відповідь: |  |  |  |  |  |  |  |  |  |  |
| 2. Agile-маніфест з'явився, коли все більше проектів почали створюватись без 100% розуміння на початку, що має бути наприкінці. В галузі створення ПЗ мало не щодня з'являються нові ідеї і вимоги від користувачів, а також нові можливості для виконання задач. Тому якщо чітко визначити і зафіксувати на етапі створення вимог і планування робіт яким саме має бути кінцевий продукт, а потім створювати його без оглядки на інші новинки, які випускаються на ринку, то к моменту випуску вашого готового продукту на ринок, він може бути вже неактуальним, тобто гроші та час будуть витрачені, а користь від продукту буде нульова. Або якщо замовнику необхідно внести зміни до вимог, яким повинен відповідати продукт, а "поїзд створення ПЗ вже в дорозі набрав швидкість", його вже дуже важко і дорого зупинити. В той час практики-розробники ПЗ вже користувались леговажними методами розробки, але їм хотілося прийти до якогось загального знаменника і просувати свої пропозиції та принципи колективно. Тоді і з'явився Agile-маніфест. Мені здається, що створення продукту за цими принципами значно полегшує, прискорює, робить більш ефективною роботу. Наприклад, не потрібно витрачати час та інші ресурси на створення та підтримку актуальності документації продукта; можна досить швидко та безболісно вносити зміни, запропоновані замовником посеред процесу, оскільки весь продукт поділений на частини, і замовник може змінювати пріоритет їх виконання, підлаштовуючись під умови ведення бізнесу, для підтримки своєї конкурентоспроможності; завдяки особистій комунікації з замовником та між членами команди підвищується взаєморозуміння, а також це сприяє озвученню та швидкому вирішенню проблем, які можуть з'явитись в процесі; крім того самоорганізована команда є більш ефективною. | | | | | | | | | | |
|  | | | | | | | | | | |
| 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. | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |
| Для випуску мобільного додатку краще скористатися гнучкою методологією Scrum, це дозволить зробити роботу швидко, без зайвої документації (в даному випадку документація не обов'язкова, оскільки функціонал додатку є нескладним), дозволить за необхідності швидко коригувати вимоги в процесі створення з урахуванням вимог ринку (можна "погратися" з випуском нових версій, враховуючи попит на цей додаток і можливі зауваження до зручності використання). Потрібно розділити продукт на маленькі частини з самого початку, і потім вибирати з беклогу найбільш приоритетні задачі на найближчий спринт і виконувати їх в визначені терміни. | | | | | | | | | | |