# 1. Склади порівняльну таблицю найбільш поширених методологій:

| № | назва методології | сильні сторони | слабкі сторони | для якої галузі є доцільною |
| --- | --- | --- | --- | --- |
| 1. | Waterfall | - із-за дуже докладної документації кожного етапу, ця модель стійка до оновлення участників. вони можуть приходити на проект і йти, коли завершать свою роботу і на терміни виконання роботи це ніяк не вплине.  - зарання відомо що буде відбуватися і на яких етапах, тому простіше прогнозувати бюджет та набирати відповідну за класифікацією команду.  - є дисципліна із-за докладної документації планов та етапів проекту. | - занадто багато документів, які треба постійно актуалізувати. поки нічого не буде согласовано і підписано - з міста нічого не здвинеться  - усі вимоги повинні бути одразу відомі. іноді замовник сам не знає чого хоче і якщо він передумає на середині створення продукту - це буде коштувати йому дорожче.  - якщо спеціалісти не дуже компетентні, вони можуть допустити помилку й із-за того, що етап тестування проводиться у кінці - помилок може назбирається дуже багато і їх виправлення буде коштувати дорого. | - доцільною буде для галузі в котрій є дуже чітка послідовність розробки, де замовник чітко знає чого він хоче, де будь яка помилка може призвести до фатальних наслідків. по цій моделі можна робити пз для космічних програм, пз для медичної апаратури, і тд. |
| 2. | V-model | - на кожному етапі відбувається контроль поточного процесу, для того, щоб переконатися в можливості переходу на наступний рівень  - тестування починається ще зі стадії написання вимог, причому для кожного наступного етапу передбачений свій рівень тестового покриття  - для кожного рівня тестування розробляється свій особистий тест план  - покращений, при порівнянні з каскадною моделлю, тайм-менеджмент | - немає можливості динамічного внесення змін  - складно виправляти фундаментні помилки. відсутній механізм вирішення проблем/помилок які були виявленні на етапі тестування  - розробка починається тільки після переходу до наступного етапу. ніяких попередніх шаблонів не передбачено. це може затруднит розуміння процесу створення пз для замовника. | підходить у тій галузі, де робота продукта не може бути зупинена. наприклад розробка ПЗ для військової техніки, авіації і тд. так як тестування починається тільки після глибокого аналізу вимог, котрі були описані в документах. |
| 3. | Ітераційна модель | - нівелювання впливу серйозних ризиків на ранніх стадіях проекту, поки це ще можна зробити з мінімальними витратами  - безперервне ітераційне тестування кінцевого продукту, що дозволяє оцінити успішність проекту в цілому  - раннє виявлення невідповідностей між вимогами, моделями і кодом - рівномірне навантаження учасників проекту  - реальна оцінка поточного стану проекту і впевненість в успішності проекту  - знижує ризики провалу проекту і розтрачання усього бюджету. | - не кожен клієнт надасть перевагу саме такому підходу, більшість буде обирати більш класичні методи  - потребує багато часу, тому можна не встигнути вкластися в дедлайн  - вимоги постійно змінюються, тому можуть бути потенційні проблеми з архітектурою і для того щоб надати їй адекватного вигляду можуть знадобитися додаткові ресурси | - підійде для тих, хто працює над невеликими стартапами та поки не знає як саме повинен виглядати готовий продукт  - для тих, хто зацікавлений у постійному розвитку свого проекту  наприклад, це можуть бути системи дистанційного навчання, месенджери, тощо.. |
| 4. | Інкрементна модель | - швидка розробка додатків  - кожен компонент доставляється клієнту після завершення  - дозволяє частково використовувати продукт ще на етапі його розробки  - дозволяє уникнути тривалого часу розробки, а також уникнути великих початкових витрат на проект та подальшого періоду очікування  - дозволяє щось змінювати у новій версії продукту  - зменшується час для тестування наступни ітерацій | - орієнтована на дуже малі збільшення функціональності  - так як ітераційна та інкрементна моделі дуже схожі то і їх недоліки також схожі. (див. стовпчик “слабкі сторони” №3. Ітераційна модель”). | - доцільна для швидкої розробки додатків  - доцільна до того ж до чого і ітераційна модель (див. стовпчик “для якої галузі є доцільною” №3. Ітераційна модель”). |
| 5. | Спіральна модель | - розбиття великого обсягу виконання робіт на невеликі частини для зручності оптимізації процесу  - можливість гнучкого проектування  - зв’язок із користувачем на ранніх стадіях розробки продукту, що забезпечує створення потрібного продукту високої якості  - можливість користувачеві приймати участь в усіх етапах розробки  - за допомогою уточнювання поставлених цілей є підвищена вирогідність передбаченої поведінки системи  - можливість регулярної оцінки затрат, яка може призвести до їх спільного скорочення | - ускладненість структури моделі, що призводить до складності її використання розробниками, менеджерами й заказчиками  - необхідність в високій кваліфікації робітників для оцінки ризиків  - можливість віддалення часу закінчення проекту у зв’язку з бажанням замовника покращувати кожну створену версію  - необхідність у чіткому розподіленні робіт між розробниками | - при розробці великих проектів.  - при розробці проектів із надто складними вимогами.  - при розробці нової серії продуктів, або систем.  - якщо замовник не впевнений у тому що саме хоче бачити  - при розробці систем, потребующих великого обсягу обчислень |
| 6. | Scrum | - адаптивність. можливість швидко вносити зміни, реагуючи на зміни на ринку чи інші можливі умови  - простий у розумінні та засвоєнні  - отримання швидких результатів, та предоставлення їх замовнику  - зв’язок із командою та замовником, що допомагає більш якісно створювати продукт та згуртувати колектив  - команда самоорганізована та функціональна  - прозорість проекту, замовник може дивитися за кожним етапом  - команда працює над різними задачами проекту одночасно, що дозволяє швидше досягти поставлених цілей | - недостатньо опрацьована документація, якщо проект потребує складної документації, треба наймати на цю роль окремого спеціаліста  - без залученості клієнта до проекта буде знижена ефективність роботи  - не усі проекти можна робити по цій методології, деяким краще підійде чіткий план  - іноді регулярні зідзвони/зустрічі із замовником можуть тормозити процес розробки із-за неотримання фідбеку  - кладно вести великі проекти, більше підходить для малих та середніх  - зміни можна вносити тільки по закінченню спринта | - більше підходить для малих та середніх проектів  - тоді, коли потрібен високий рівень контролю над виконанням роботи  - коли продукт потрібно швидко вдосконалювати |
| 7/ | Kanban | - потоковий робочий процес  - щоденні зустрічі можуть бути плюсом, якщо добре організовані і не затримують процес роботи  - внесення змін у будь-який момент, в порівнянні із скрамом, де зміни можна вносити тільки по закінченню спринту.  - немає часових обмежень. ти робиш поки не зробиш  - підвищення ефективності роботи. оскільки співробітники не чекають, поки їм дадуть роботу, а відразу по завершенню завдання беруть інше, немає простою  - краща взаємодія в команді. члени команди завжди в курсі того, хто чим займається та як рухається проєкт — все це є на дошці.  - дуже гнучка метедологія.  - ніяких офіційних ролей не існує, тому її легко підлаштувати під структуру вашої компанії і можна перебудувати в будь-який момент під конкретні потреби  - менше часу витрачається на планування і більше на роботу, опрацювання певних завдань для робітників | - процес може загальмувати навіть якщо раніше усе йшло добре кожен у команді повинен чітко знати що саме він потрібен робити. мають бути добре відпрацьовані певні процеси  - вимагає постійної наявності завдань, по типу “конвеєра”. якщо вони закінчаться, почнеться процес руйнування  - можуть бути проблеми з дедлайнами проекту із-за відсутності часових обмежень, отже, це може бути й мінусом | - для проектів, котрим потрібно змінюватися  - вважаю, що так як і канбан і скрам дуже сході то й галцзь для котрої вони будуть доцільні - однакові (див. стовпчик “для якої галузі є доцільною”. №6. Scrum.”). |

# 

# 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?

## 1. На твою думку, чому з’явився Agile-маніфест?

спочатку були тільки послідовні моделі розробки ПЗ, розробники та науковці працювали над чимось новим і винайшли гнучкі моделі. тобто ітеративні підходи вже існували, на момент створення аджайл. але людство хотіло чогось більше, розвивати сферу гнучких методологій й далі, зробити щось, що не потребувало б дуже довгого планування та супер докладної документації, те, що допомогло б випускати нові версії продукту швидше, щоб замовник бачив процес розробки. вони хотіли зробити так, щоб люди в команді більше комунікували один з одним, підвищували свою продуктивність та працездатність, хотіли мати можливість вносити зміни та корективи в будь яку секунду. тому в лютому 2001 року був опублікований Agile-манифест, групою іменувавшою себе «Agile Alliance».

Цінностями цієї методології стали такі вислови:  
1. Люди та взаємодія важливіші за процеси та інструменти.

2. Працюючий продукт важливіший за вичерпну документацію.

3. Співпраця із замовником важливіша за узгодження умов контракту.

4. Готовність до змін важливіша за проходження початкового плану.

### 2. Які проблеми він мав вирішити і чи це вдалося?

Аджайл мав вирішити такі проблеми, як:

1. бистріше починати розробку проекту, вичислюваючи сроки та ризики проекту під час розробки
2. більше контакту із клієнтом/замовником. бистріше віддавати продукт клієнту, зібрати отзиви та почати допрацьовувати продукт
3. работа зі змінюючимися потребами к продукту на всьому терміні роботи над проектом

приклад роботи з Аджайл:



якщо повертатися до питання “чи вдалося Аджайл маніфесту вирішити ті проблеми, котрі він перед собою поставив” то відповідь - ТАК. Звісно вдалося. зараз аджайл найпопулярніша методологія розробки ПЗ та найзручніша. поділяється на Scrum, Kanban та ScrumBan.

# 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?

якщо це стартап, я б обрала щось із гнучкої методології, таке як Scrum, ScrumBan чи Kanban. бо мої бачення проекту можуть змінюватися, плавати і, можливо, мені сподобається якась пропозиція моєї команди що до покращення застосунку. у мене є просто ідея “мобільний застосунок для обміну світлинами котиків” і я чітко ще не знаю як саме хочу бачити функціонал та самий додаток, я хочу мати можливість вносити корективи на будь якому етапі розробки мого ПЗ. плюсом до всього можуть бути мітинги, це допоможе нашій команді комунікувати та “жити” нашим проектом.

саме тому ці методології здаються мені доречними.