**SDLC та STLC. Методології розробки ПЗ.**

Перший рівень

1. **Waterfall** - кожна наступна стадія починається тільки після того, як закінчилася попередня. Якщо все робити правильно, то «водоспад» буде найшвидшою та найпростішою моделлю.

* розробку легко контролювати. Замовник завжди знає, чим зараз зайняті програмісти, може керувати термінами та вартістю;
* вартість проекту визначається на початковому етапі. Усі кроки заплановані вже на етапі узгодження договору;
* не потрібно наймати тестувальників із серйозною технічною підготовкою. Тестувальники можуть спиратися на детальну технічну документацію;
* тестування розпочинається на останніх етапах розробки. Якщо у вимогах до продукту припустилися помилки, то виправити її коштуватиме дорого. Тестувальники виявляють її, коли вже написаний код та документація;
* замовник бачить готовий продукт наприкінці розробки і лише тоді може дати зворотний зв'язок. Велика ймовірність того, що результат його не влаштує;
* не підходить для складних та великих проектів;
* не підходить, там, де вимоги постійно змінюються.

«Водоспад» підходить для розробки проектів у медичній та космічній, атомній галузі, де вже сформовано велику базу документів (стандартів та специфікацій), на основі яких можна написати вимоги до нового ПЗ.

**V-образна модель** - це вдосконалена каскадна модель, у якій замовник із командою програмістів одночасно пишуть вимоги до системи та описують, як тестуватимуть її на кожному етапі.

* кількість помилок в архітектурі програмного забезпечення зводиться до мінімуму;
* строга етапність;
* планування, тестування та верифікація системи відбувається на ранніх стадіях;
* недостатня гнучкість;
* якщо при розробці архітектури припустилися помилки, то повернутися і виправити її коштуватиме дорого, як і в «водоспаді»;
* недостатній аналіз ризиків;
* нема можливості динамічного внесення змін.

V-модель підходить для проектів, у яких важлива надійність та ціна помилки дуже висока. Наприклад, при розробці подушок безпеки для автомобілів або систем спостереження за пацієнтами у клініках.

**Spiral model** - використовуючи цю модель, замовник та команда розробників серйозно аналізують ризики проекту та виконують його ітераціями. Наступна стадія ґрунтується на попередній, а наприкінці кожного витка – циклу ітерацій – приймається рішення, чи продовжувати проект.

* уникнення ризиків - оскільки велика увага приділяється їх опрацюванню;
* вимоги більш точні;
* швидкий розвиток;
* є ризик застрягнути на початковому етапі - безкінечно удосконалюючи першу версію продукту;
* розробка триває довго і коштує дорого.

Спіральна модель може використовуватися для наукових та дослідницьких проектів.

Другий рівень

1. +
2. Agile з'явився для вирішення таких потреб:

* коли потрібно розробити інноваційний продукт;
* коли треба побудувати чи вдосконалити процес, у якому є висока частка невизначеності;
* коли ми працюємо із замовником із розмитими потребами.

Використовуючи Agile-філософію ми часто спілкуємося із замовником демонструючи йому проміжні результати та цікавимося побажаннями для вдосконалення чи додатку нового функціоналу. Вже на проміжних етапах замовник може користуватися продуктом і таким чином якнайскоріше отримувати результат. Agile дозволяє досягати безпрецедентної гнучкості. Члени команди працюють разом, виконуються невеликі спринти та постійна підтримка зворотного зв'язку. Усі власники продуктів почуваються досить залученими до проектів.

Agile чудово працює у стартапах, маленьких командах, але на великих проектах, як правило, призводить до великих проблем: відстрочок запуску на місяці і навіть повного провалу.

При розробці Agile - в продакшн відправляються невеликі зміни. Багів у таких оновленнях міститься значно менше.

На перший погляд, чудово, проте насправді роботи у Quality Assurance реально додається, за такого навантаження служба не справляється з тестуванням — пропускає серйозні помилки, не встигає перевірити роботу системи загалом. Так як автоматичні тести розраховані на перевірку окремих частин - а не продукту вцілому.

Третій рівень

1. +
2. Для створення мобільного застосунку для обміну світлинами котиків я б обрала Kanban, тому що на сьогодні це одна з найпопулярніших методологій розробки ПЗ. Команда веде роботу за допомогою віртуальної дошки, яка розбита на етапи проекту. Кожен учасник бачить, які завдання перебувають у роботі, які застрягли на одному з етапів, а які вже дійшли до його стовпця і потребують уваги. На відміну від скраму, в канбані можна взяти термінові завдання на розробку відразу, не чекаючи початку наступного спринту. Канбан зручно використовувати та можливо наочно відстежувати прогрес.