ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

C#. Розробка ПЗ

Лекція 16

План

1. Життєвий цикл програмного забезпечення
2. Моделі ЖЦПЗ
   - Каскадна модель (Waterfall Model)
   - Інкрементна модель (Incrementing Model)
   - Ітеративна модель (Iterative Model)
   - Спіральна модель (Spiral Model)
   - V-модель (v-model)
   - Гнучка модель (Agile Model)
   - SCRAM

Життєвий цикл програмного забезпечення

Життєвий цикл ПЗ (ЖЦПЗ) – це безперервний процес, який починається з моменту прийняття рішення про необхідність створення програмного забезпечення (ПЗ) і закінчується в момент його повного вилучення з експлуатації.

Життєвий цикл програмного забезпечення

Основним нормативним документом, який регламентує ЖЦ, є міжнародний стандарт ISO/IEC 12207. Він визначає його структуру, містить процеси, дії і задачі, які повинні бути виконані під час створення ПЗ. Структура ЖЦ за стандартом ISO/IEC 12207 базується на трьох групах процесів:

- основні процеси (придбання, постачання, розробка, експлуатація, супровід);
- допоміжні процеси (документування, управління конфігурацією, забезпечення якості, перевірка, атестація);
- організаційні процеси (управління проектами, створення інфраструктури проекту, визначення, оцінка і покращення самого життєвого циклу, навчання).

Моделі життєвого циклу ПЗ

Модель ЖЦ – структура, яка визначає послідовність виконання та взаємозв’язок процесів, дій та задач на протязі виконання проекту.

Модель ЖЦ залежить від специфіки програмного продукту, а також від специфіки умов, в яких даний продукт створюється та функціонує.

Моделі життєвого циклу ПЗ

Існує два основних типи моделей ЖЦ:

Прогнозовані моделі ЖЦ – в основі цих моделей лежить чітке планування усіх стадій процесу розробки ПЗ.

Адаптивні моделі ЖЦ (так звані гнучкі технології) – їх особливістю є сприйняття та адаптація процесів розробки під потреби замовника.

Моделі життєвого циклу ПЗ

На даний час найпоширенішими серед прогнозованих моделей є:

- Каскадна модель;
- V-подібна модель;
- Інкрементна модель;
- Спіральна модель;
- Модель швидкої розробки;
- Модель еволюційного прототипування.

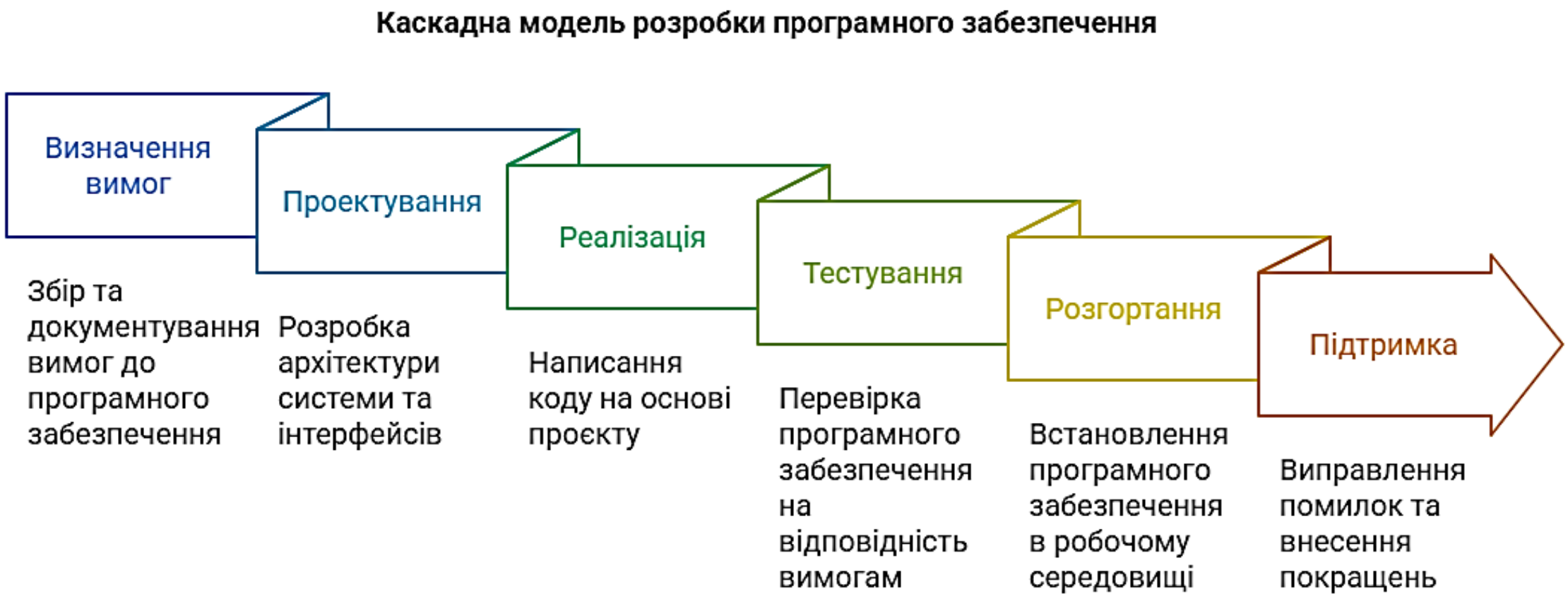
Моделі життєвого циклу ПЗ

Найвідомішими адаптивними моделями ЖЦ є:

- Scrum;
- XP (Extreme Programming);
- Adaptive Software development (ASD);
- Dynamic System Development Model (DSDM);
- Feature Driven Development (FDD)

Адаптивні моделі ЖЦ включають не тільки опис структури фаз, та їх пояснення, але і рекомендації, підходи їх ефективного використання.

Каскадна модель (Waterfall Model)



Каскадна модель (Waterfall Model)

Стисла характеристика:

- *фіксований набір стадій;*
- *кожна стадія закінчується документованим результатом;*
- *наступна стадія починається лише після закінчення попередньої.*

Недоліки:

- *негнучкість;*
- *фаза повинна бути завершена до переходу до наступної;*
- *набір фаз фіксований;*
- *важко реагувати на зміни вимог.*

***Використання**: там, де вимоги добре зрозумілі та стабільні.*

Інкрементна модель (Incrementing Model)

Інкрементна модель передбачає розробку програмного забезпечення серією невеликих, самостійних циклів (ітерацій). Кожна ітерація включає:

- планування,
- проектування,
- реалізацію,
- тестування,
- оцінку.

Результат кожної ітерації є частково готовим продуктом, який поступово вдосконалюється.

Інкрементна модель (Incrementing Model)

Переваги:

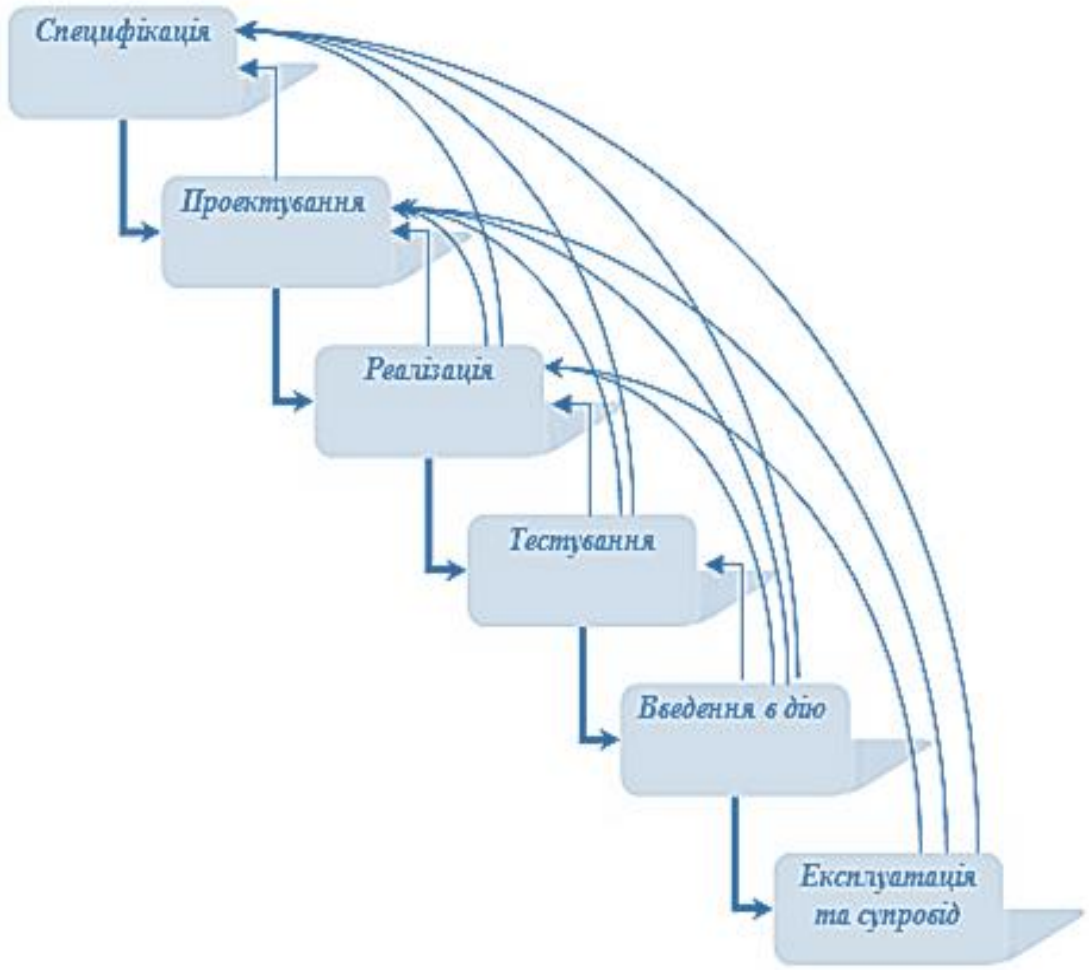
- *Є можливість раннього виходу на ринок, щоб подивитися реакцію користувачів*
- *Базова версія ПЗ коштує дешевше, а модулі можна дороблять в міру появи грошей або не робити зовсім через непотрібність*
- *Найризикованіші ідеї можна відкласти на потім*
- *Виправлення помилок обходиться дешевше*

Недоліки:

- *Вимоги до проєкту на кожному етапі мають бути чітко визначені та зрозумілі*
- *Необхідний хороший менеджмент*
- *Застосунок може вийти занадто “сирим” і не дожити до появи всіх функцій*

Ітеративна модель (Iterative Model)

Ітеративна модель починається з простої реалізації підмножини вимог до програмного забезпечення та ітеративного вдосконалення версії програмного забезпечення доки не буде реалізовано весь продукт, який перебуває в розробці. На кожній ітерації вносяться зміни в дизайн і додаються нові функціональні можливості. Основна ідея цього методу виникає в тому, щоб розробити систему через повторювані цикли (ітераційні) і невеликими порціями за раз.

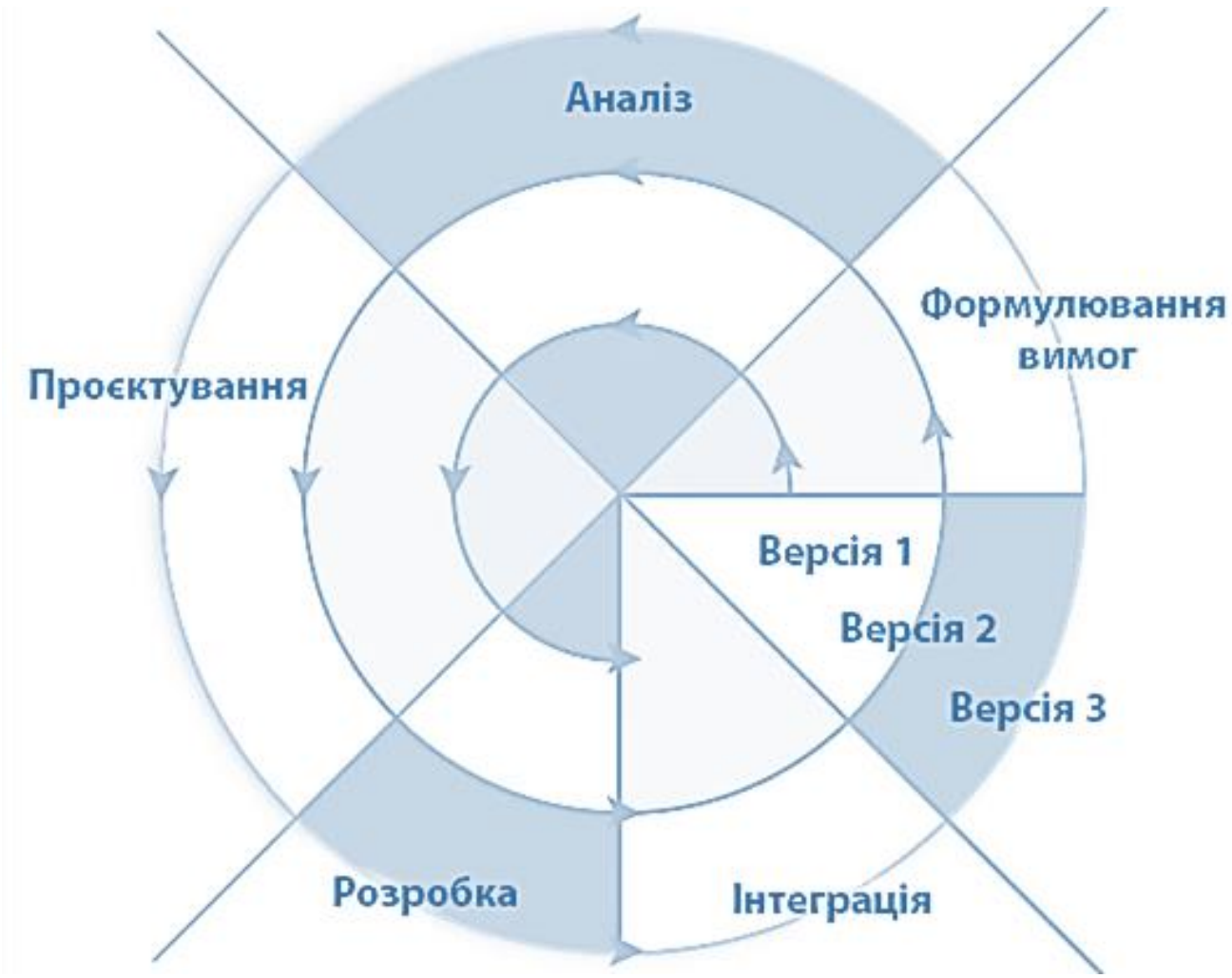


Ітеративна модель (Iterative Model)

Використання:

- *Якщо вимоги до системи чітко розписані та всім зрозумілі.*
- *Проєкт за своїм обсягом дуже великий.*
- *Головну мету визначено, але деталі реалізації можуть змінюватись у процесі роботи.*

Спіральна модель (Spiral Model)



Спіральна модель поєднує елементи каскадної та інкрементної моделей з акцентом на аналізі ризиків.

Кожна фаза розробки представлена у вигляді спіралі, де кожен виток представляє ітерацію. Кожна ітерація включає чотири основні види діяльності: планування, оцінка ризиків, інжиніринг (розробка та тестування) та оцінка результатів.

Спіральна модель (Spiral Model)

Переваги:

- *приділяється особлива увага управлінню ризиками;*
- *додаткові функції можуть бути додані на пізніх етапах;*
- *є можливість гнучкого проєктування.*

Недоліки:

- *оцінка ризиків на кожному етапі є досить витратною;*
- *постійні відгуки і реакція замовника може провокувати все нові і нові ітерації, які можуть призводити до тимчасового затягування розробки продукту;*
- *більш застосовується для великих проєктів.*

V-модель (v-model)

V-модель – це покращена версія класичної каскадної моделі, де на кожному етапі відбувається контроль поточного процесу, для того, щоб переконатися в можливості переходу на наступний рівень.

У цій моделі тестування починається ще зі стадії написання вимог, причому для кожного наступного етапу передбачений свій рівень тестового покриття.

V-модель (v-model)

Переваги:

- *Тестування проходить на всіх етапах розробки*
- *Імовірність помилок зводиться до мінімуму*

Недоліки:

- *Потрібен високий рівень кваліфікації тестувальників та їхня висока зайнятість*
- *Якщо помилки все ж таки пропустили, то повернутися до попереднього етапу буде навіть дорожче, ніж на каскадній моделі*

Використання: *якщо програмний продукт потребує скрупульозного тестування, для невеликих та середніх проєктів, з чітко визначеними вимогами.*

Гнучка модель (Agile Model)

Гнучка модель (Agile Model) – являє собою сукупність різних підходів до розробки ПЗ. Включає серії підходів до розробки програмного забезпечення, орієнтовані на використання ітеративної розробки, динамічне формування вимог і забезпечення їх реалізації в результаті постійної взаємодії всередині самоорганізованих робочих груп, що складаються з фахівців різного профілю.

Окрема ітерація являє собою мініатюрний програмний проєкт. Однією з основних ідей Agile – є взаємодія всередині команди та з замовником обличчя до обличчя, що дозволяє швидко приймати рішення і мінімізує ризики розробки програмного забезпечення.

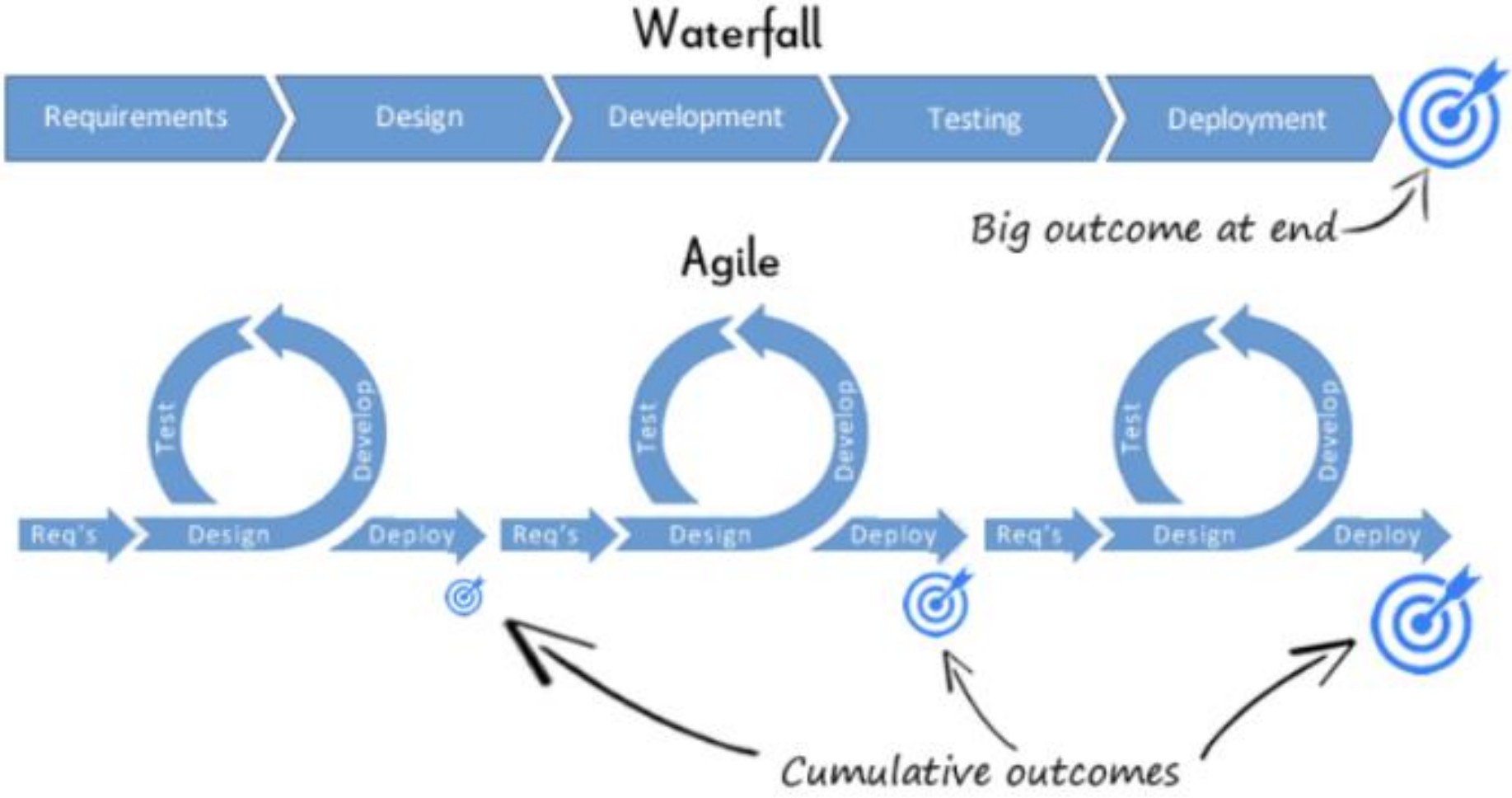
Гнучка модель (Agile Model)

Переваги:
- *швидке прийняття рішень завдяки постійним комунікаціям;*
- *мінімізація ризиків;*
- *полегшена робота з документацією.*

Недоліки:
- *велика кількість мітингів і обговорень, що може збільшити час розробки продукту;*
- *складно планувати процеси, так як вимоги постійно змінюються;*

Використання: *рідко використовується для реалізації великих проєктів.*

Waterfall Model - Agile Model



SCRAM

Скрам - це гнучка модель розробки ПЗ, в якій робиться акцент на якісному контролі процесу розробки.

Ролі в методології (Scrum Master, Product Owner, Team) дозволяють чітко розподілити обов'язки в процесі розробки. За успіх Scrum в проєкті відповідає Scrum Master і є сполучною ланкою між менеджментом і командою. За розробку продукту відповідає Product Owner, який також ставить завдання і приймає остаточні рішення для команди.
Команда - це єдине ціле, в ній результати оцінюються не по кожному окремому учаснику, а по тому, що виходить в результаті у всіх.
Спринти в даній методології тривають від 1 до 4 тижнів. Після кожного спринту команда надає варіант закінченого продукту.

SCRAM

Переваги:

- *швидкий зворотній зв'язок від фахівців в різних сферах (дизайнерів, архітекторів, тестувальників та ін.);*
- *завдяки залученості тестувальника в роботу відбувається швидке додавання нового функціоналу і швидкий запуск продукту з мінімальними функціями;*
- *самостійна і самоорганізована команда.*

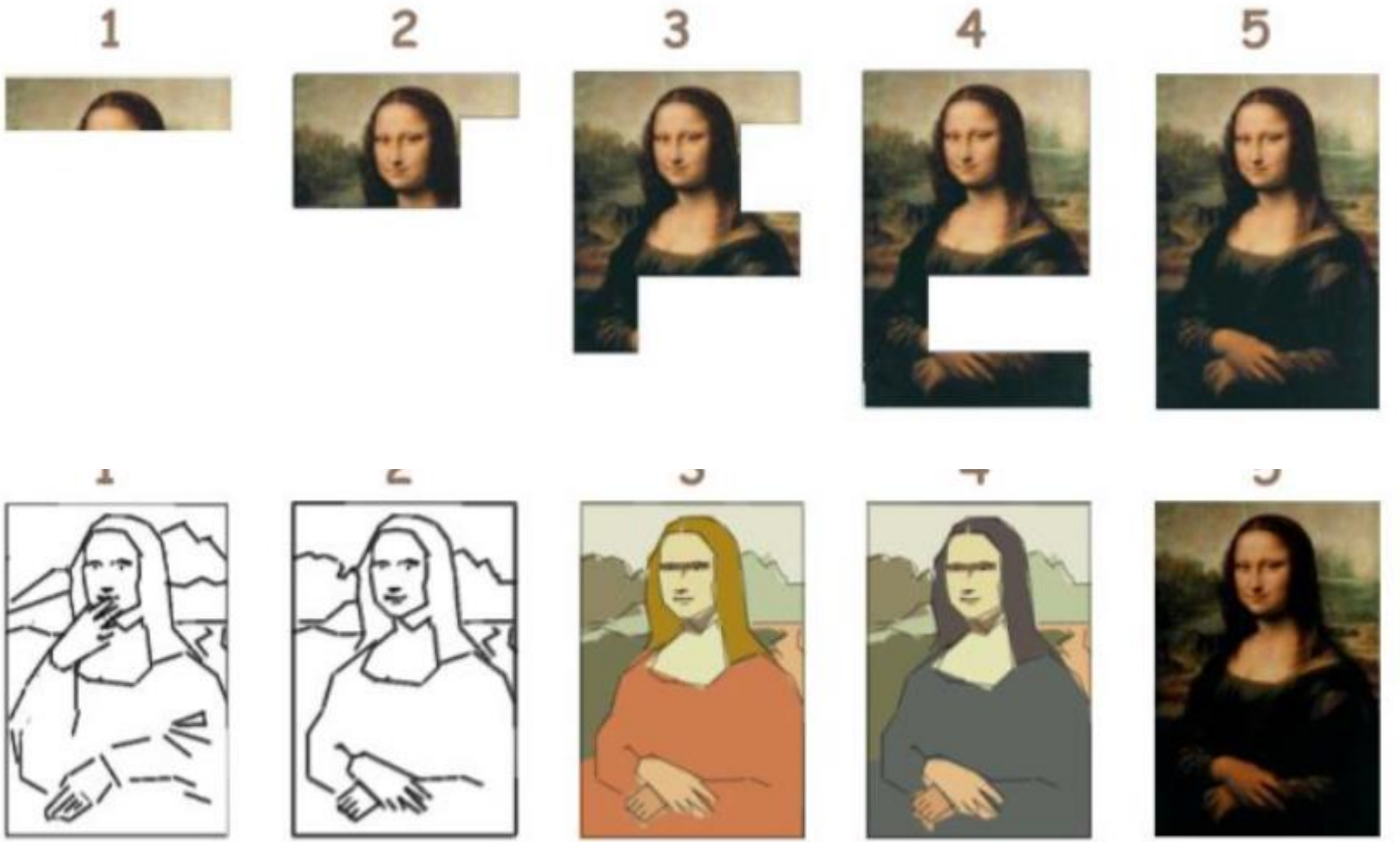
Недоліки:

- *деякі люди, які знають продукт, стають незамінними, так як документація не надається в процесі розробки;*
- *неможливо спланувати точну дату завершення, так як все уточнюється за результатами попереднього спринту;*
- *замовники не завжди можуть зрозуміти суть даної методології і необхідно витратити час на "лікбез".*

Каскадна модель

Автор зображень, Джефф Паттон, обґрунтував гнучкий метод роботи у своїй статті 2007 року « Не знаю, чого хочу, але знаю, як цього досягти ».

Інкрементна і ітеративна модель



Agile Model - SCRAM