Тема 6. Життєвий цикл ІС (SDLC)

Життєвий ІС

- Кожна система має свій життєвий цикл.
- Життєві цикли розрізняються за властивостями, цілям, використання системи, а також по переважаючих умовах. Проте, не дивлячись на очевидні безліч відмінностей в життєвих циклах систем, існує базовий набір стадій життєвого циклу, що становлять повний життєвий цикл будь-якої системи.
- Кожна стадія має певну мету і внесок в повний життєвий цикл і розглядається при плануванні і виконанні життєвого циклу системи.
- Життєвий цикл інформаційної системи характеризується періодом часу від ідеї створення інформаційної системи і закінчуючи моментом виведення її з експлуатації
- SDLC (Software Development Life Cycle) — структурований процес створення програмного забезпечення.

Етапи SDLC

1) Планування
2) Аналіз вимог
3) Проектування
4) Розробка
5) Тестування
6) Впровадження
7) Підтримка

Етапи SDLC. Планування

- Визначення цілей і масштабів проєкту.
- Прийняття рішень щодо ресурсів і технологій.
- Формування попереднього бачення архітектури.

Етапи SDLC. Аналіз вимог

- Збір і документування вимог
- Виявлення функціональних та нефункціональних вимог
- Узгодження з архітектурними обмеженнями

Етапи SDLC. Проектування

- Вибір архітектурного стилю (client-server, SOA)
- Моделювання компонентів та інтеграцій

Етапи SDLC. Розробка

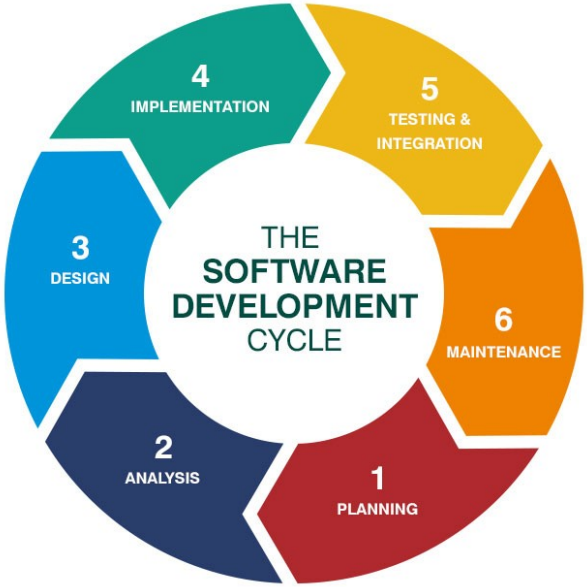
- Реалізація системи відповідно до архітектури.

Етапи SDLC. Тестування

- Перевірка відповідності реалізації вимогам.
- Архітектурні тести (продуктивність, безпека, масштабованість)

Етапи SDLC. Тестування

- Перевірка відповідності реалізації вимогам.
- Архітектурні тести (продуктивність, безпека, масштабованість)



Етапи SDLC. Впровадження

- Розгортання системи у робочому середовищі.
- Налаштування інфраструктури.
- Міграція даних.

Етапи SDLC. Підтримка

- Моніторинг та усунення помилок.
- Оптимізація продуктивності.
- Внесення змін при еволюції системи.

Моделі SDLC

- Waterfall
- V-Model
- Spiral
- Iterative
- Agile

Модель життєвого циклу розробки програмного забезпечення (SDLC) концептуально представляє SDLC організованим способом, щоб допомогти організаціям реалізувати його.

Різні моделі мають фази SDLC в різному хронологічному порядку для оптимізації циклу розробки. Нижче ми розглянемо деякі популярні моделі SDLC.

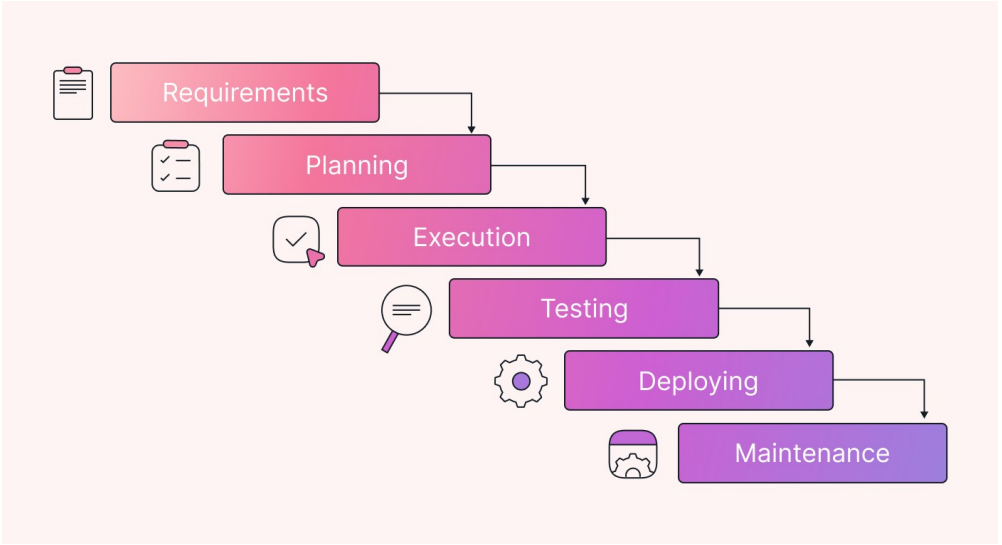
Вибір моделі впливає на процес архітектурного проектування

Каскадна модель SDLC (Waterfall)

У каскадній моделі всі етапи влаштовані послідовно, так що кожен новий етап залежить від результатів попереднього. Концептуально розвиток переходить від однієї фази до іншої, як каскад.

Переваги та недоліки

Каскадна модель забезпечує дисципліну в управлінні проектами і дає відчутний результат в кінці кожного етапу. Однак після того, як етап вважається завершеним, залишається мало місця для змін, оскільки зміни можуть вплинути на час доставки, вартість і якість програмного забезпечення. Тому модель найбільш підходить для невеликих проектів з розробки програмного забезпечення, де завдання легко організувати і контролювати, а вимоги можна точно визначити заздалегідь.

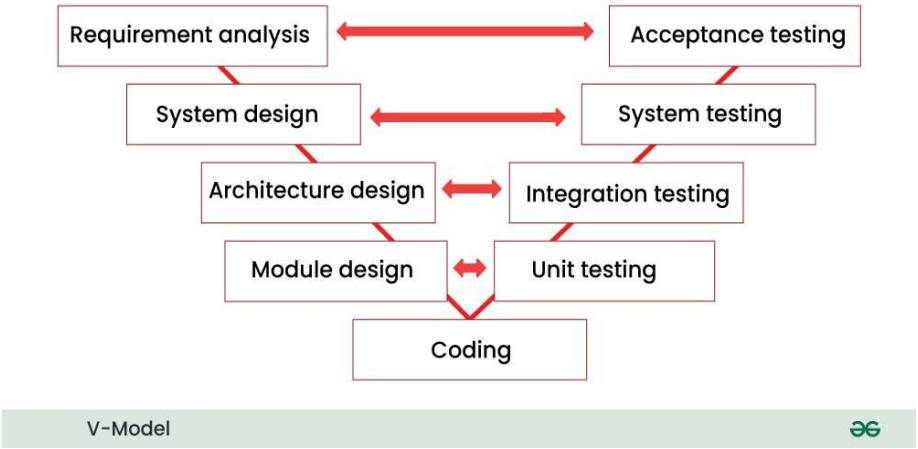


V-Model SDLC

У V-моделі процес розробки також організований поетапно, але на відміну від каскадної моделі, вона підкреслює тісний зв’язок між етапами розробки та етапами тестування. Кожна стадія створення продукту має свій відповідний етап перевірки: наприклад, вимоги співвідносяться з приймальним тестуванням, проєктування — з інтеграційним, а кодування — з модульним. Завдяки цьому процес візуально утворює форму літери “V”.

Переваги та недоліки

V-модель забезпечує раннє планування тестування та підвищує якість кінцевого продукту за рахунок чіткої відповідності між вимогами й перевірками. Вона зручна для проектів, де можна заздалегідь чітко визначити вимоги і є високі стандарти якості. Водночас модель менш гнучка: внесення змін на пізніх етапах потребує повернення на попередні рівні, що може бути витратним за часом і ресурсами. Тому V-модель найкраще підходить для проєктів з чіткими вимогами та високими ризиками, наприклад у сферах із жорсткими стандартами (медицина, авіація).

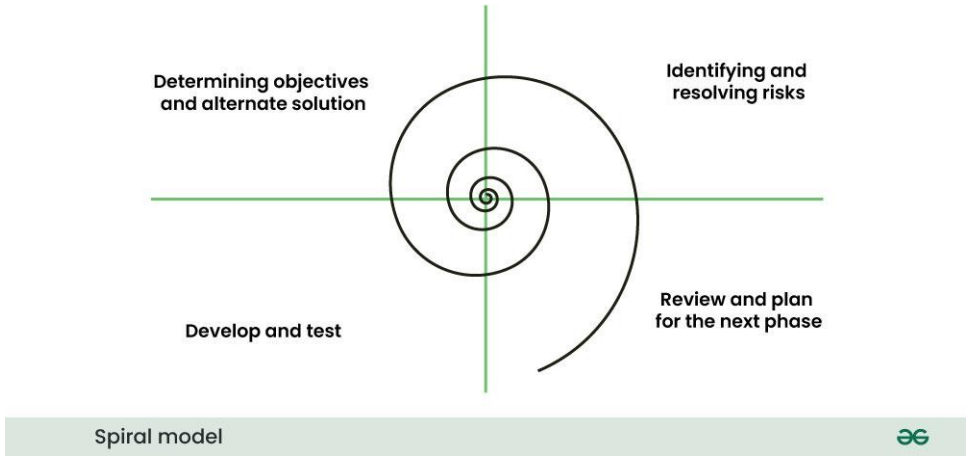


Спіральна модель SDLC (Spiral model)

Спіральна модель поєднує в собі невеликі повторювані цикли ітераційної моделі з лінійним послідовним потоком каскадної моделі для визначення пріоритетів аналізу ризиків. Ви можете використовувати спіральну модель для забезпечення поступового випуску та вдосконалення програмного забезпечення, створюючи прототипи на кожному етапі.

Переваги та недоліки

Спіральна модель підходить для великих і складних проектів, які вимагають частих змін. Однак це може дорого коштувати невеликим проектам з обмеженим масштабом.

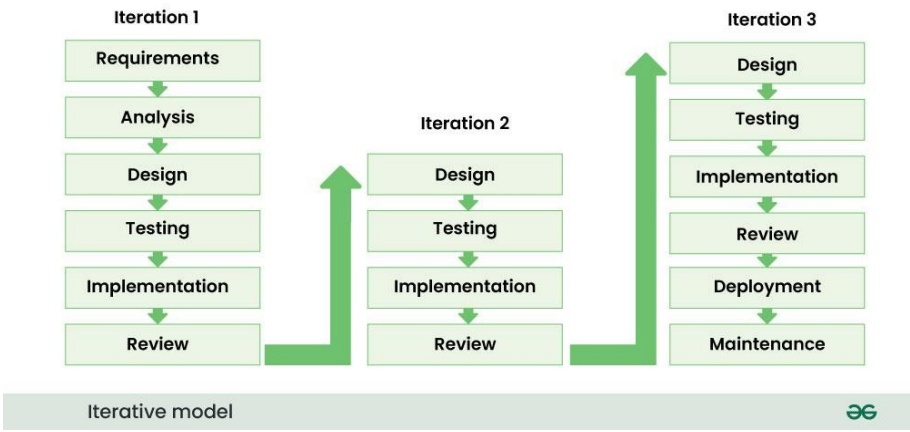


Ітеративна модель SDLC (Iterative)

Ітераційний процес передбачає, що команди починають розробку програмного забезпечення з невеликою підмножиною вимог. Потім поступово вдосконалюють версії, поки програмне забезпечення не буде готове до виробництва. В кінці кожної з ітерацій команда створює нову версію програмного забезпечення.

Переваги та недоліки

Легко виявляти та керувати ризиками, оскільки вимоги можуть змінюватися між ітераціями. Однак повторювані цикли можуть призвести до зміни обсягу робіт і недооцінки ресурсів.

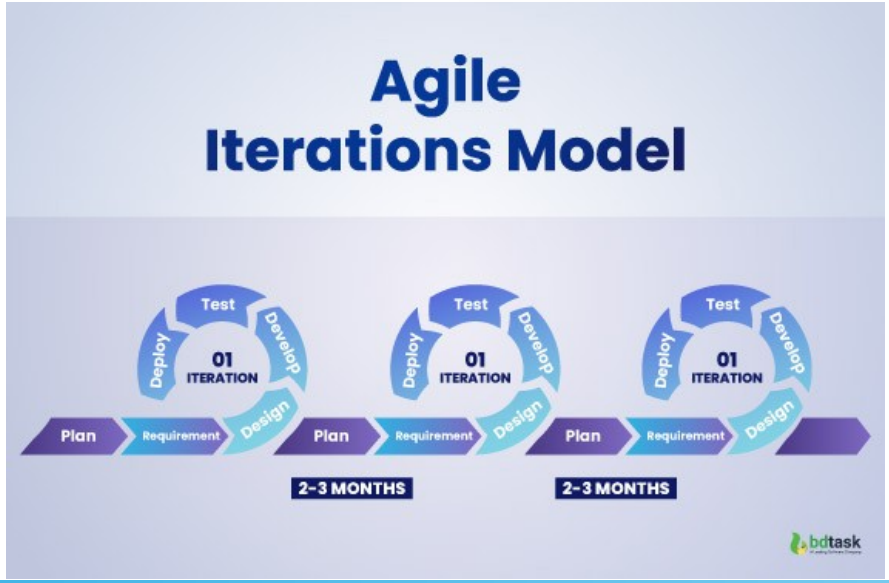


Гнучка модель SDLC (Agile)

У гнучкій моделі кроки SDLC поділяються на кілька циклів розробки. Команда швидко проходить всі етапи ітерацій, вносячи лише невеликі додаткові зміни в кожному циклі. Фахівці постійно оцінюють вимоги, плани і результати, щоб швидко реагувати на зміни. Гнучка модель є ітераційною та поступовою, що робить її більш ефективною, ніж інші моделі процесів.

Переваги та недоліки

Швидкі цикли розвитку допомагають командам виявляти та вирішувати проблеми в складних проектах на ранніх стадіях, перш ніж вони стануть серйозними. Вони також можуть залучити клієнтів та зацікавлених сторін, щоб отримати зворотний зв'язок протягом усього життєвого циклу проекту. Однак надмірна залежність від відгуків клієнтів може призвести до надмірної зміни обсягу робіт або завершення проекту на півдорозі.



Висновки

- SDLC і архітектура ІС тісно взаємопов’язані.
- Архітектурні рішення приймаються на кожному етапі.
- Якісна архітектура = успішний SDLC.

Дякую за увагу!