Управління проектами

ДЕНЬ 7

Супровід (продовження…)

Документація супроводу

SLA

Інструкції

Процеси

Життєвий цикл розробки

Життєвий цикл розробки

Життєвий цикл розробки (SDLC – Software Development LifeCycle) – це період часу, що складається з фаз, який починається з моменту задуму системи і закінчується, коли система більше не доступна для використання

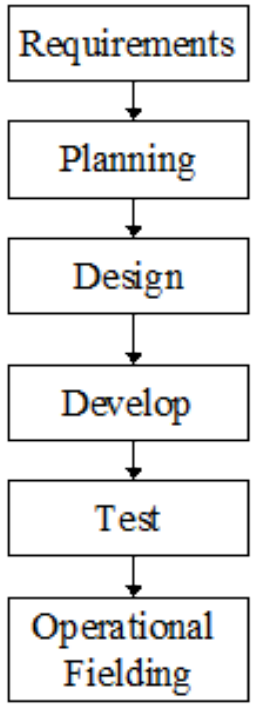
Базові моделі життєвого циклу

водоспадна

інкрементна

еволюційна

Водоспадна модель



Водоспадна модель.2

У цьому підході кожен процес виконується послідовно, і кожен процес завершується перед переходом до наступного процесу в послідовності

Наприклад, аналіз і проектування не починаються, доки плани проекту не будуть підготовлені, розглянуті та завершені

Аналогічно, розробка не починається, поки вимоги та етапи проектування не будуть завершені.

Водоспадна модель.3

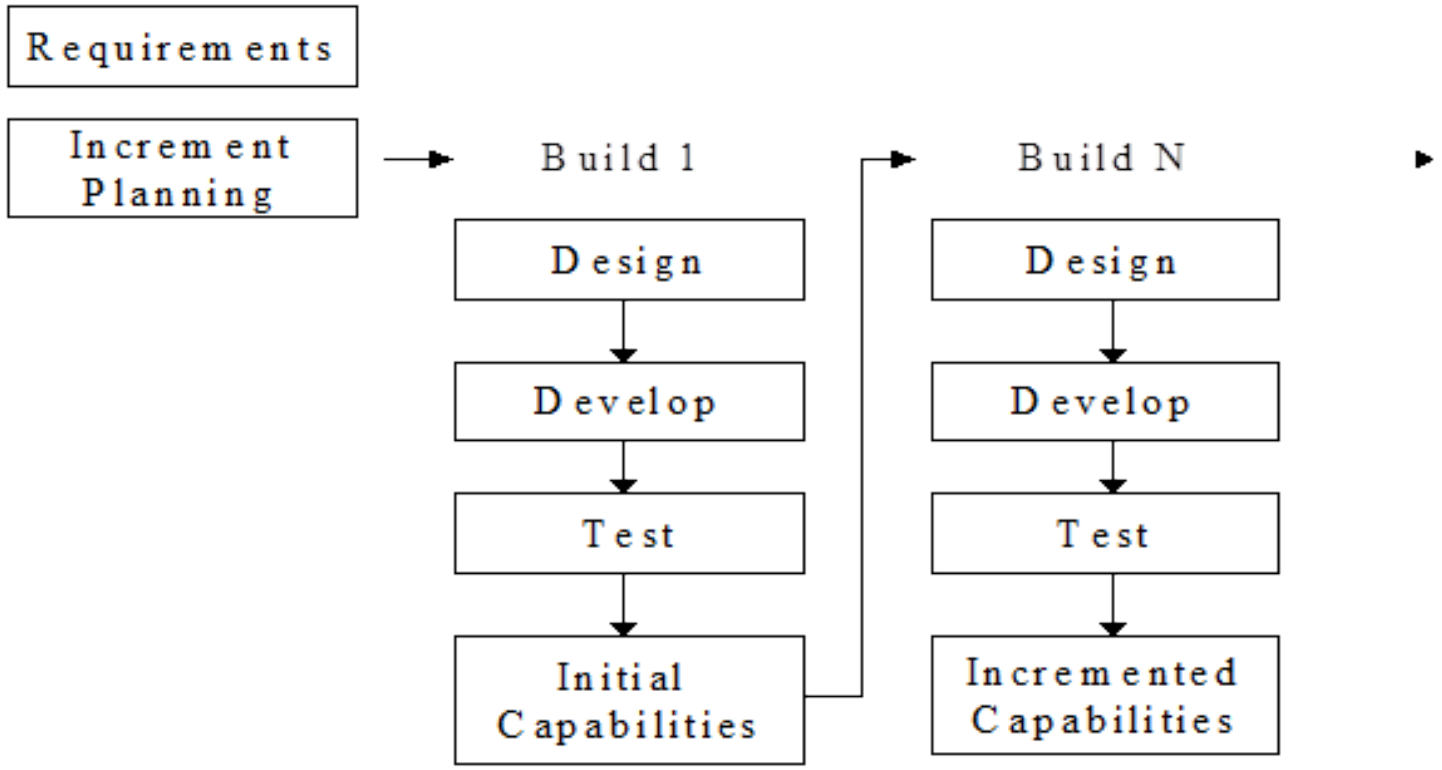
Стратегія забезпечує структурований, дисциплінований метод розробки програмного забезпечення та може бути корисною для проектів технічного обслуговування та невеликих нових стартів із чітко визначеними та зрозумілими вимогами

Однак для інших типів проекту ця стратегія може виявитися ризикованою та негнучкою

Для більших систем, які мають лише один прохід через процес, проблеми інтеграції часто виникають занадто пізно в процесі розробки, і готовий продукт недоступний до самого кінця процесу

Тривалий період між початком проекту та постачанням продукту може перешкодити залученню клієнтів і призвести до процесу розробки системи, на який постійно впливають зміни вимог клієнтів

Інкрементна модель



Інкрементна модель.2

Інкрементна стратегія (також відома як «попередньо запланованe покращення продукту») передбачає поділ системи на кілька «білдів» (або інкрементів) і розробку системи по одному випуску за раз

Команда виконує планування проекту та аналіз вимог лише один раз, а потім повторює процеси проектування, розробки та тестування для кожної збірки системи

Перша збірка системи включає підмножину запланованих можливостей; наступна збірка додає ще одну підмножину запланованих можливостей, поки прирости не утворюють повну систему

Інкрементна модель.3

Інкрементна стратегія найбільш підходить для великих нових систем, де вимоги до системи та програмного забезпечення можуть бути повністю визначені та чітко зрозумілі

Основною перевагою цієї стратегії перед водоспадною стратегією є використання кількох циклів розробки

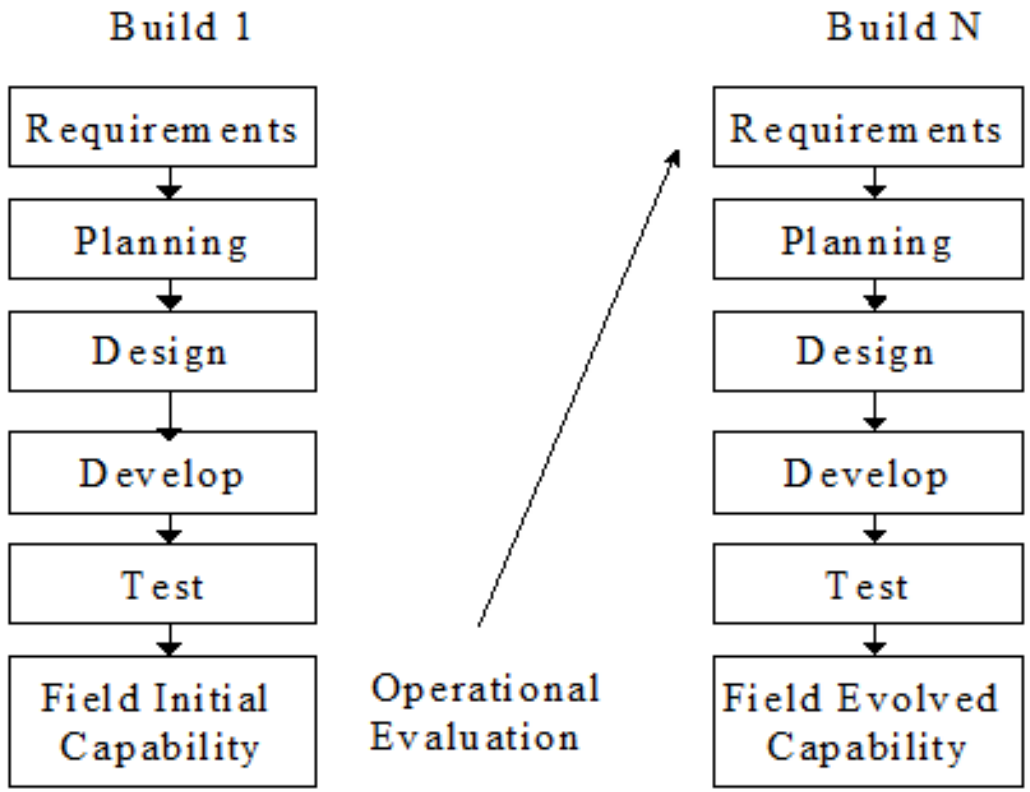
Це дозволяє підійти до розробки з більш керованими кроками

Крім того, у спонсора є можливість набагато швидше взаємодіяти з реальною системою та надавати зворотній зв’язок розробникам

Основним недоліком стратегії Incremental є її залежність від наявності чітко і повністю визначених вимог до програмного забезпечення на початку

Проблеми виникають у разі швидко мінливих вимог клієнтів

Еволюційна модель



Еволюційна модель.2

Еволюційна стратегія також розробляє систему в збірках, але відрізняється від інкрементної стратегії визнанням того, що користувач не повністю розуміє всі вимоги наперед

Основна відмінність полягає в тому, що проект, що використовує еволюційну стратегію, повторює фазу аналізу вимог більше одного разу і створює програму на послідовних рівнях повноти, і кожен рівень є версією програми, яку певною мірою можна використовувати

Як і у випадку з інкрементною стратегією, проект проходить через кілька циклів розробки і створює кілька збірок

Перша створена збірка — це операційна версія, яка відповідає початковому набору функціональних, системних та програмних вимог

На основі відгуків клієнтів проект повторює процеси аналізу, проектування, розробки, тестування та впровадження, щоб створити другу версію, яка відповідає більш чітко визначеним функціональним, системним та програмним вимогам

Цей процес триває до тих пір, поки вимоги не будуть повністю визначені та зрозумілі і не буде створена остаточна система

Еволюційна модель.3

Стратегії еволюційної моделі особливо підходять для ситуацій, коли, хоча загальна природа програми відома, детальні вимоги до системи та програмного забезпечення важко дізнатися, визначити або охарактеризувати

Зазвичай це відбувається з програмними системами підтримки прийняття рішень, які є високоінтерактивними та мають складні людино-машинні інтерфейси

До недоліків використання еволюційної стратегії можна віднести те, що клієнти/користувачі можуть передчасно прийняти одну з версій як остаточну систему; і проект легко зазнає надмірного зростання вимог, що дозволяє додатковим і розширюючим вимогам відкласти або підвищити вартість розробки

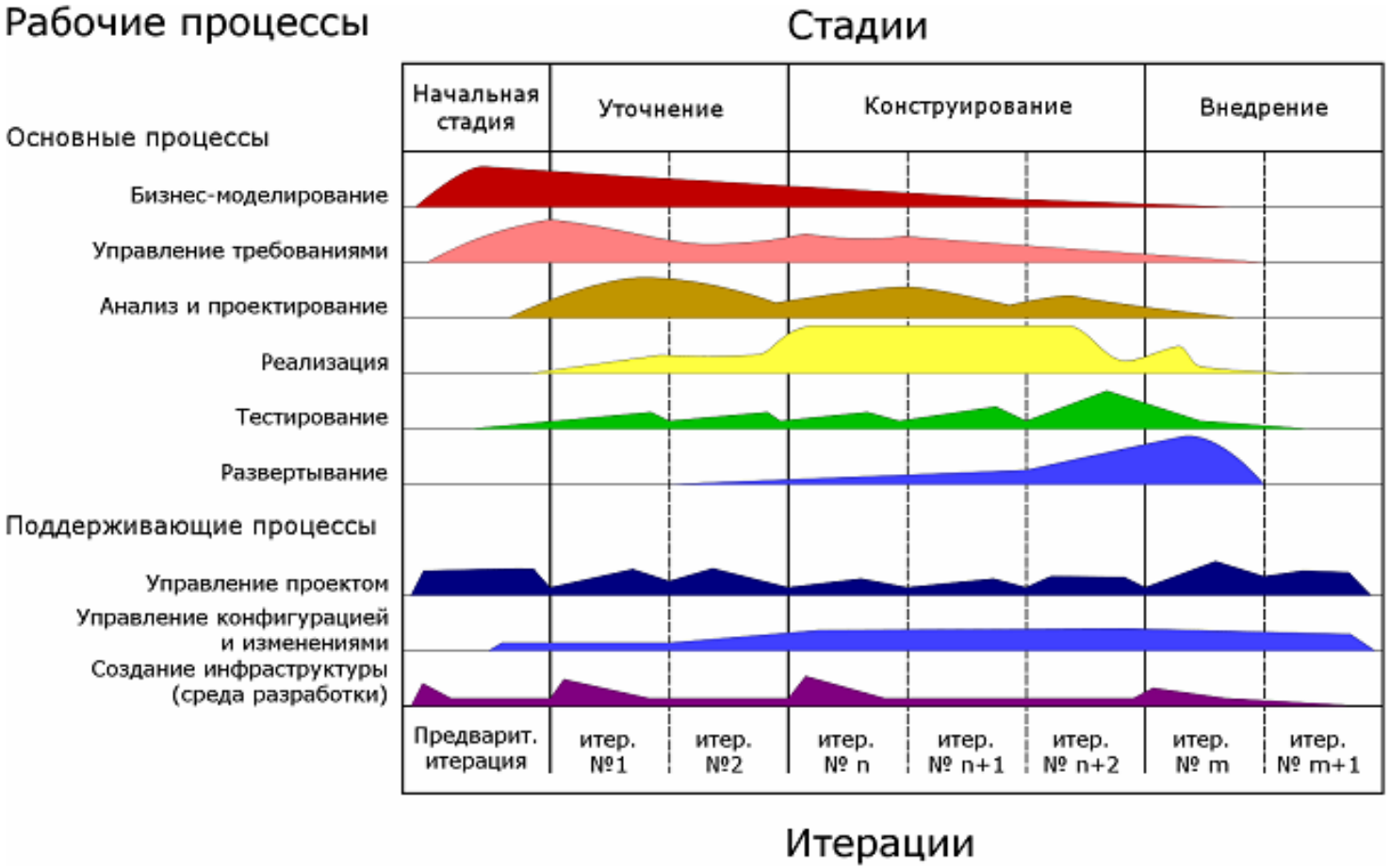
Фактори вибору моделі життєвого циклу

Factor	Opportunity	Risk
1. Requirements not well clear	Evolutionary	Waterfall, Incremental
2. System too large to do once	Incremental, Evolutionary	Waterfall
3. Full capability needed at once	Waterfall	Incremental, Evolutionary
4. Part Capability needed early	Incremental, Evolutionary	Waterfall
5. Old system reengineering	Incremental, Evolutionary	Waterfall
6. Rapid changes in requirements anticipated	Evolutionary	Waterfall, Incremental
7. Rapid changes in technologies anticipated	Evolutionary	Waterfall, Incremental
8. Long-run staff/commitment doubtful	Waterfall	Incremental, Evolutionary

Життєвий цикл за ГОСТ 34.*

1. Формування вимог
2. Проектування
3. Реалізація
4. Тестування
5. Введення в дію
6. Експлуатація та супровід

Життєвий цикл за RUP



Життєвий цикл від MSF

Життєвий цикл від SCRUM