ЛЕКЦІЯ 2
МОДЕЛІ ЖИТТЄВОГО ЦИКЛУ ПРОЕКТУ

План лекції

Вступ

1. Поняття життєвого циклу програмного проекту
2. Прогнозовані моделі життєвого циклу
   - 2.1. Особливості каскадної моделі розробки проекту
   - 2.2. V-подібна модель життєвого циклу
   - 2.3. Модель еволюційного прототипування (Макетування)
   - 2.4. Модель швидкої розробки (Еволюційна модель)
   - 2.5. Інкрементна модель (Incremental model)
   - 2.6. Спіральна модель
3. Огляд гнучких моделей життєвого циклу (Agile, Lean, Scrum, Kanban, RUP, DSDM, RAD, XP)

Вступ

Важливою рисою будь-якого проекту є процес його виконання. Процес передбачає розтягнутість у часі і має також назву життєвий цикл, що більш точно відповідає таким обов'язковим складовим проекту як його дата початку і кінця.

Однак менеджер проекту не може створювати кожного разу життєвий цикл «з нуля», замість цього він звертається до перевіреної і узагальненої практики, яка адаптується для конкретного проекту. Для полегшення проєктування, створення і випуску якісного програмного продукту існують різні моделі життєвого циклу проекту.

1. Поняття життєвого циклу програмного проекту

Модель життєвого циклу проекту (Software life cycle model, SLCM) – це структура, що визначає послідовність виконання та взаємозв'язку процесів, дій та задач на протязі виконання проекту. Тобто модель ЖЦ розробки ПЗ схематично пояснює яким чином будуть виконуватися дії з розробки програмного продукту, шляхом опису "послідовності" цих дій. Така послідовність може бути або не бути лінійною, оскільки фази можуть слідкувати одна за одною, повторюватися або відбуватися послідовно.

Існує багато моделей життєвого циклу, які з'являлися і розвивалися поступово. Менеджер проекту має розуміти переваги і недоліки кожної з них, для того, щоб обрати і використати правильну модель для кожного конкретного проекту.

Ефективне управління процесом розробки ПЗ можливе за умов повного розуміння процесів, що відбуваються в процесі роботи над проектом.

Життєвий цикл проекту представляє собою модель виконання проекту, від рівня відповідності якої підходам, що використовуються для управління, у значній мірі залежить кінцевий успіх проекту.

На початкових етапах розробки ПЗ процес не був структурованим, його модель можна було представити як постановку завдання та його виконання до того часу, доки воно не буде зроблено, чи відмінено.

Можливість використання такої моделі відповідала характеру ПЗ для якого вона використовувалася – це, насамперед, ПЗ, що створювалося для наукових розрахунків. Подібне ПЗ не відрізнялося структурною складністю, було порівняно невеликим за обсягами.

Однак використання неструктурованої моделі життєвого циклу для великих і складних за структурою програмних проектів ускладнено, оскільки модель не відповідає процесам, що необхідні для створення подібного ПЗ.

Існує два основних типи моделей життєвого циклу:

1. Прогнозовані моделі життєвого циклу – в основ цих моделей лежить чітке планування усіх стадій процесу розробки програмного забезпечення.

2. Адаптивні моделі життєвого циклу (так звані гнучкі технології) – особливістю цих моделей є сприйняття та адаптація процеси розробки під потреба замовника та ринку.

На сьогоднішній момент найбільшого поширення серед прогнозованих набули такі моделі:

- каскадна;
- V-подібна модель;
- модель прототипування;
- модель швидкої розробки;
- інкрементна модель;
- спіральна модель.

Адаптивні моделі життєвого циклу з’явилися досить недавно, але здобули вже досить широке поширення особливо серед ,багатьох компаній. Найвідомішими є

- Agile model
- Lean
- Scrum
- Extreme Programming, XP
- Adaptive Software development, ASD
- Dynamic System Development Model, DSDM
- Feature Driven Development, FDD

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

2.1. Особливості каскадної моделі(Waterfall) розробки проекту

Каскадна модель життєвого циклу була розроблена Вінстоном Ройсом в 1970 році та вперше представлена в його книжці “Управління розробкою великих програмних систем”.

Характерною рисою каскадної моделі можна назвати те, що вона представляє собою формальний метод, різновид розробки “зверху-вниз”, вона складається з незалежних фаз, що виконуються послідовно –рис. 1.

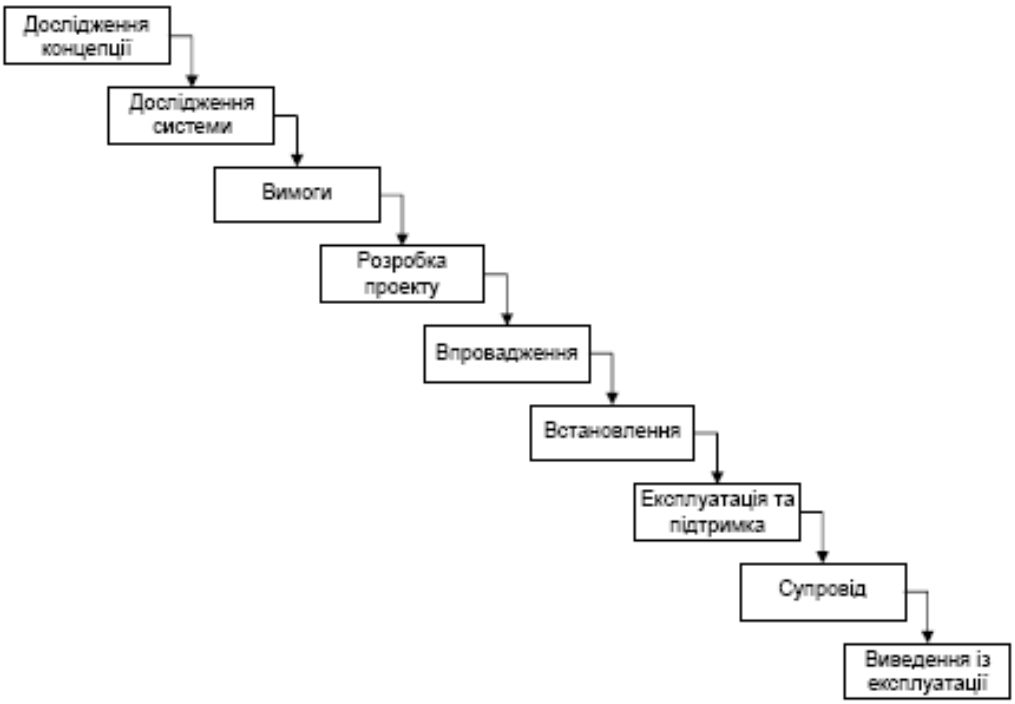

Рис. 1. Каскадна модель життєвого циклу проекту

В моделі передбачено, що кожна наступна фаза починається лише тоді коли повністю завершено виконання попередньої фази. Кожна фаза має певні критерії входу та виходу: вхідні та вихідні дані. В результаті виконання генеруються внутрішні та зовнішні дані проекту, включаючи документацію та ПЗ.

Розглянемо детальніше фази каскадної моделі:

1. дослідження концепції – відбувається дослідження вимог на системному рівні ціллю визначення можливості реалізації концепції;

2. процес системного розподілення – може бути пропущений для систем, що стосуються виключно розробки програмних продуктів. Для систем,

в яких необхідна розробка як апаратного, так і програмного забезпечення, необхідні функції застосовуються до програмного продукту і обладнання у відповідності з загальною архітектурою системи;

3. процес визначення вимог – визначаються програмні вимоги до інформаційної предметної області системи, призначення, лінії поведінки, продуктивності та інтерфейси. В разі необхідності в процес також включено функціональний розподіл системних вимог щодо апаратного і програмного забезпечення

4. процес розробки проекту – розробляється і формулюється логічно послідовна технічна характеристика програмної системи, включаючи структури даних, архітектуру програмного забезпечення, інтерфейси представлення та алгоритмічну деталізацію

5. процес реалізації – в результаті виконання цієї фази ескізний опис ПЗ перетворюється в повноцінний програмний продукт. При цьому створюється вод, баз даних та документація, що лежать в основі фізичного перетворення проекту. Якщо програмний продукт являє собою придбаний пакет прикладних програм, то основними діями щодо його реалізації є установка та тестування пакету програм. Якщо програмний продукт розробляється на замовлення, то основними діями є програмування та тестування

6. процес установки – включає в себе установку ПЗ, його перевірку та офіційне затвердження замовником.

7. процес експлуатації та підтримки – дана фаза включає в себе запуск користувачем системи та поточну підтримку, що включає в себе надання технічної допомоги, обговорення виниклих у користувача питань, реєстрація запитів користувача щодо модернізації та внесення змін, а також коректування та усунення помилок

8. процес супроводу – дана фаза пов’язана з усуненням помилок, недоліків, з ітерації розробки і передбачає зворотний зв’язок для надання інформації щодо встановлених дефектів.

9. процес виводу з експлуатації – виведення існуючої системи з її активного використання шляхом припинення її роботи, або завдяки її заміни новою системою або модернізованою версією вже існуючою системи;

10. інтегральні задачі – включають початок роботи над проектом, моніторінг проекту та його управління, управління якістю, верифікацію та атестацію, менеджмент конфігурації, розробку документації та професійну підготовку на протязі всього життєвого циклу.

Основна суть моделі Waterfall у тому, що етапи залежать один від одного і наступний починається, коли завершений попередній, утворюючи таким чином поступальний (каскадний) рух уперед. Команди різних етапів між собою не комунікують, кожна команда відповідає чітко за свій етап.

Каскадна модель набула дуже широкого поширення і використовується на практиці вже більш ніж 30 років. За всі ці роки використання дана модель здобула багато прихильників і багато противників.

Основні переваги даної системи:

1) дана модель є простою та зрозумілою, вимогою для успішного завершення програмного продукту є виконання запланованих дій; фази моделі гарно визначені;

2) каскадна модель дозволяє учасникам проекту, що закінчили свої фази, брати участь в інших проектах;

3) зручна для використання менеджером проекту: сприяє здійсненню строго контролю над перебігом проекту, полегшує роботу над розробкою проектного плану;

4) забезпечує гарний контроль над якістю проекту. Усі закінчені фази та отримані дані підлягають огляду. Ця процедура забезпечує визначення якості програмної системи.

Недоліки каскадної моделі життєвого циклу

1) складність чіткого формулювання вимог на початку життєвого циклу і неможливість їх динамічної зміни на його протязі;

2) послідовність лінійної структури процесу розробки, в результаті повернення до попередніх кроків для вирішення виникаючих проблем призводить до збільшення витрат і порушення графіка робіт;

3) непридатність проміжного продукту для використання;

4) неможливість гнучкого моделювання систем, що не мають аналогів;

5) пізнє виявлення проблем, пов’язаних зі складанням, у зв’язку з одночасною інтеграцією всіх результатів в кінці розробки;

6) недостатня участь користувача у створенні системи – тільки на самому початку (при розробці вимог) і в кінці (під час приймальних випробувань);

7) неможливість попередньої оцінки якості системи користувачем;

8) проблемність фінансування проекту, пов’язана зі складністю одноразової розподілу великих грошових коштів.

Область застосування каскадної моделі.

Через недоліки каскадної моделі її застосування необхідно обмежувати ситуаціями, в яких вимоги та їх реалізація максимально чітко визначені та зрозумілі.

- при розробці проектів з чіткими, незмінними протягом ЖЦ вимогами, зрозумілими реалізацією і технічними методиками;
- при розробці проекту, орієнтованого на побудову системи або продукту такого ж типу, як вже розроблялися розробниками раніше;

• при розробці проекту, пов’язаного зі створенням і випуском нової версії вже існуючого продукту або системи;

• при розробці проекту, пов’язаного з перенесенням вже існуючого продукту на нову платформу;

• при виконанні великих проектів, в яких задіяно декілька великих команд розробників.

Якщо компанія має досвід побудови визначеного виду систем - автоматизованого бухгалтерського обліку, нарахування зарплати тощо - тоді в проекті, орієнтованому на побудову ще одного продукту того ж типу, можливо навіть з використанням вже існуючих розробок, то в цьому випадку можна ефективно використовувати каскадну модель,

2.2. V-подібна модель життєвого циклу

На заміну каскадній моделі була запропонована V-подібна (шарнірна) модель, яка покликана усунути недоліки каскадної моделі, зокрема поєднати етапи постановки завдання та перевірки його відповідності специфікаціям (рис. 2):

Рис. 2. V-подібна модель

Суть цієї моделі полягає в тому, що процеси на всіх етапах контролюються, щоб переконатися в можливості переходу на наступний рівень. Вже на стадії написання вимог починається процес тестування.

Незважаючи на те, що передбачається план тестування розробляти на ранніх етапах ЖЦПЗ, саме тестування здійснюється вже після того, як продукт створено.

Отже, помилки, виявлені на цьому етапі (пов’язані з неправильним проектуванням) спричинять повернення на попередні етапи життєвого циклу, а отже призведуть до значних фінансових втрат

Плюси:

сувора етапізація;

мінімізація ризиків і усунення потенційних проблем за рахунок того, що тестування з’являється на самих ранніх стадіях;

вдосконалений тайм-менеджмент.

Мінуси:

неможливість адаптуватися до змінених вимог замовника;

тривалий час розробки (іноді триває до декількох років) призводить до того, що продукт може бути вже не потрібний замовнику, оскільки його потреби змінюються;

немає дій, спрямованих на аналіз ризиків.

Проте, незважаючи на всі свої переваги та спрямованість на тестування, V-подібна модель має послідовну структуру: кожна наступна фаза починається лише по завершенню попередньої.

2.3. Модель прототипування (макетування)

Досить часто користувач-замовник формулює тільки загальні цілі програмного продукту і не визначає усі вимоги. З іншого боку, розробник може сумніватися в сумісності продукту з ОС, формі діалогу з користувачем, або в ефективності реалізованого алгоритму. В цьому випадку розробка прототипу програмного продукту часто виявляється найкращою моделлю процесу розробки.

Головна мета створення прототипу – уточнення вимог користувача і перевірка основних ідей проектованого продукту. У ряді випадків прототип дозволяє визначити здійсненність проекту. *Макетування, або прототипування,* – це процес створення моделі потрібного програмного продукту.

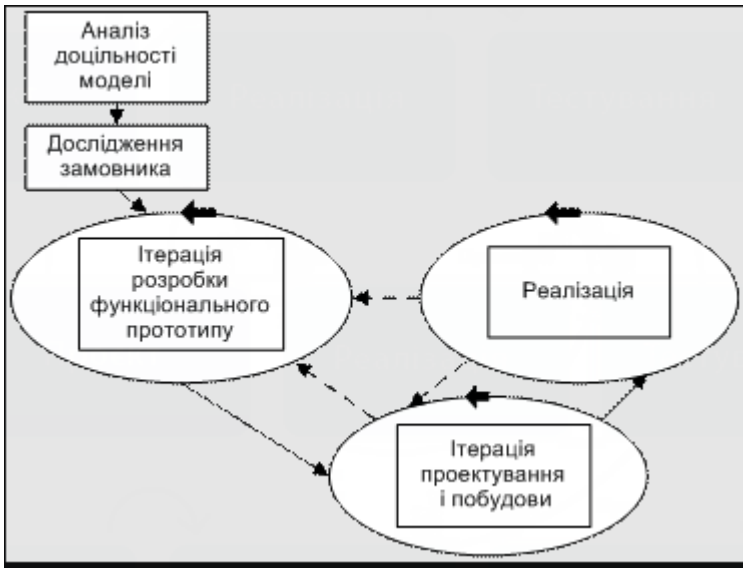

Рис. 3. Етапи моделі прототипів

Макетування базується на багаторазовому повторенні ітерацій, в який беруть участь замовник і розробник. Воно починається зі збору й уточнення вимог до створюваного ПЗ. Розробник і замовник зустрічаються і визначають всі цілі ПЗ, встановлюють, які вимоги відомі, а які потрібно конкретизувати. Потім виконується швидке проектування, при якому зосереджуються на тих характеристиках ПЗ, які повинен бачити користувач. Швидке проектування приводить до побудови макету, який оцінюється замовником і використовується для уточнення вимог до ПЗ. Ітерації повторюються до тих пір, поки макет не виявить всі вимоги замовника і, тим самим, не дасть можливості розробнику зрозуміти, що потрібно робити.

За результатами визначення прототипу розробник надає замовнику готовий прототип, а користувач замовник оцінює рівень повноти виконання заданих функцій і підтверджує застосування прототипу або відхиляє.

Переваги:

1. Безпосередній зв'язок із замовником на етапі затвердження прототипу дозволяє звести до мінімуму кількість невідповідностей вимогам програмного продукту.
2. У процесі розробки завжди можна врахувати нові вимоги замовника.
3. Замовник бачить прогрес в процесі розробки, що створює позитивний імідж розробника.
4. Ймовірність виникнення непорозумінь між замовником та розробником значно зменшується.

Недоліки:

1. Розв’язок складних задач у процесі проектування на майбутнє.

2. Замовник може зупинитись на певному прототипі, що в свою чергу не дозволить розробнику отримати більш якісний програмний продукт.

3. Знаходження прототипу може безпідставно бути розтягнуто в часі.

4. На початку робіт важко передбачити кількість ітерацій, що потрібно буде використати, що ускладнює процес планування і визначення остаточних термінів завершення проекту.

2.4. Модель швидкої розробки (Еволюційна модель)

У разі еволюційної моделі (в літературі вона часто називається моделлю швидкої розробки програм RAD (Rapid Application Development)) система послідовно розробляється з блоків конструкцій. На відміну від інкрементної моделі в еволюційній моделі вимоги встановлюються частково і уточнюються в кожному наступному проміжному блоці структури системи.

Використання еволюційної моделі припускає проведення дослідження предметної області для вивчення потреб її замовника і аналізу можливості застосування цієї моделі для реалізації. Модель використовується для розробки нескладних і некритичних систем, де головною вимогою є реалізація функцій системи.

Рис.4. Модель швидкої розробки

Переваги:

•Застосування сучасних інструментальних засобів дозволяє значно скорочувати термін розробки.

•Залучення до роботи замовника значно знижує ризики того, що він буде незадоволений готовим програмним засобом.

•За цією моделлю можливе багаторазове застосування компонентів уже існуючих програм.

Недоліки:

•У разі, якщо замовник не в змозі постійно приймати участь у процесі

розробки, то це може негативно відбитись на ПП.

•Для виконання такого проекту потрібні висококваліфіковані програмісти, які не тільки можуть виконувати своє завдання, а і користуватись сучасними інформаційними засобами.

•Існує ризик, що робота над програмним продуктом може ніколи не завершитись внаслідок відсутності домовленості про результат між замовником та розробником, в цьому випадку потрібно всім вміти вчасно зупинитись.

2.5. Інкрементна модель (Incremental model)

Відповідно до інкрементної моделі (англ. Increment – збільшення, прирощення) ЖЦ проекту розбивається на послідовність ітерацій, кожна з яких нагадує "міні-проект", включаючи всі процеси розробки в застосуванні до створення менших фрагментів функціональності, порівняно з проектом в цілому. Мета кожної ітерації – отримання працюючої версії програмної системи, що включає функціональність, визначену інтегрованим змістом всіх попередніх та поточної ітерації. Результат фінальної ітерації містить всю необхідну функціональність продукту. Таким чином, із завершенням кожної ітерації продукт отримує приріст - інкремент – до його можливостей, які, значить, розвиваються еволюційно. З точки зору структури ЖЦ таку модель називають ітеративною. З точки зору розвитку продукту – інкрементальною.

Вимоги до системи визначаються на самому початку роботи, після чого процес розробки проводиться у вигляді послідовності версій, кожна з яких є закінченим і працездатним продуктом (рис. 5).

Рис.5. Інкрементна модель

При такому підході кожна версія повинна бути придатною для використання і забезпечувати деяку підмножину необхідних функцій. Кожна лінійна послідовність виробляє працюючий інкремент ПЗ. Перший інкремент приводить до отримання базового продукту, який реалізує базові вимоги (більшість додаткових вимог залишаються нереалізованими). План наступного

інкременту передбачає модифікацію базового продукту, яка забезпечить додаткові характеристики і функціональність.

Плюси:

замовник може дати свій відгук щодо кожної версії продукту;

є можливість переглянути ризики, які пов'язані з витратами і дотриманням графіка;

звикання замовника до нової технології відбувається поступово.

Мінуси:

функціональна система повинна бути повністю визначена на початку життєвого циклу для виділення ітерацій;

при постійних змінах структура системи може бути порушена;

терміни здачі системи можуть бути збільшені через обмеженість ресурсів (виконавці, фінанси).

2.6. Спіральна модель

Спіральна модель (рис. 6) була вперше сформульована Баррі Боем в 1988 р. Відмінною особливістю цієї моделі є спеціальна увага ризикам, що впливає на організацію життєвого циклу.

Головне досягнення спіральної моделі полягає в тому, що вона пропонує спектр можливостей адаптації вдалих аспектів існуючих моделей процесів життєвого циклу.

У спіральній моделі життєвий шлях продукту, що розробляється зображується у вигляді спіралі, яка, розпочавшись на етапі планування, розкручується з проходженням кожного наступного кроку. Таким чином, на виході з чергового витка отримуємо готовий протестований прототип, який доповнює існуючу збірку. Прототип, що задовольняє всі вимоги, готовий до випуску.

Дана модель життєвого циклу характерна при розробці новаторських (нетипових) систем. На початку роботи над проектом у замовника і розробника немає чіткого бачення підсумкового продукту (вимоги не можуть бути чітко визначені) або стовідсоткової впевненості в успішній реалізації проекту (ризики дуже великі). В зв'язку з цим приймається рішення розробки системи по частинах з можливістю зміни вимог або відмови від її подальшого розвитку. Як видно з рис. 6, розвиток проекту може бути завершено не тільки після стадії впровадження, але і після стадії аналізу ризику.

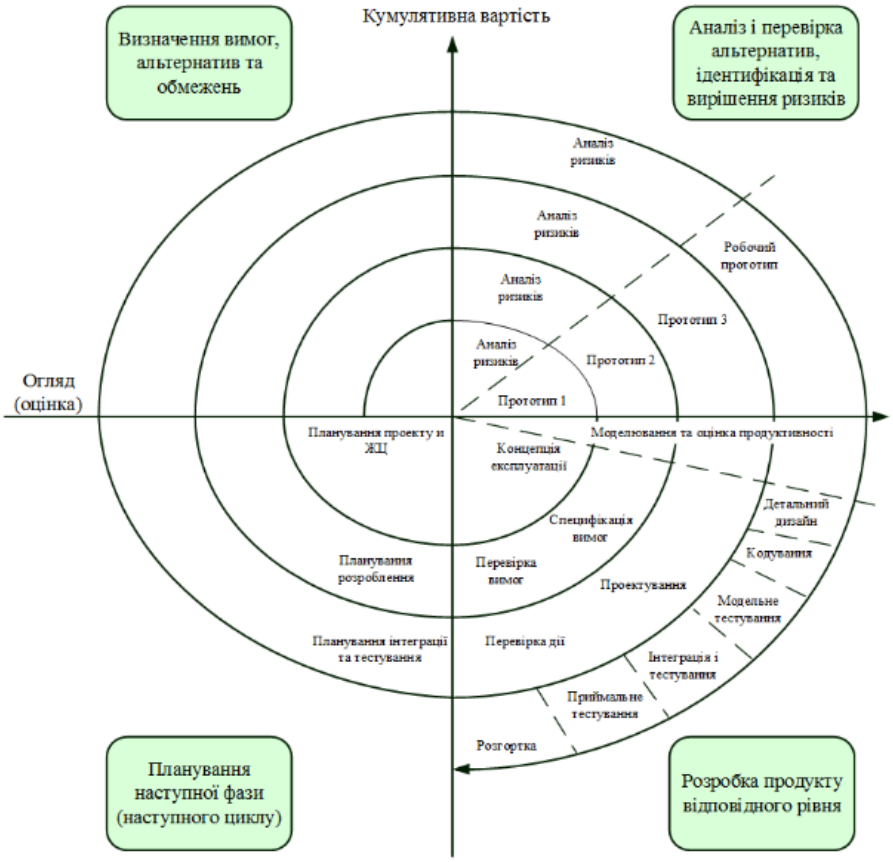

Рис. 6. Оригінальна спіральна модель ЖЦ, розроблена Боемом

Переваги моделі:

- Дозволяє швидше показати користувачам системи працездатний продукт, тим самим, активізуючи процес уточнення і доповнення вимог;
- Допускає зміну вимог при розробці інформаційної системи, що характерно для більшості розробок, у тому числі і типових;
- Забезпечує більшу гнучкість в управлінні проектом;
- Дозволяє отримати більш надійну і стійку систему. По мірі розвитку системи помилки і слабкі місця виявляються і виправляються на кожній ітерації;
- Дозволяє удосконалювати процес розробки – аналіз, проведений в кожній ітерації, дозволяє проводити оцінку того, що має бути змінено в організації розробки, і поліпшити її на наступній ітерації;
- Зменшуються ризики замовника. Замовник може з мінімальними для себе фінансовими втратами завершити розвиток неперспективного проекту.

Недоліки моделі:

- Збільшується невизначеність у розробника в перспективах розвитку проекту. Цей недолік випливає з попереднього достоїнства моделі;
- Ускладнені операції тимчасового і ресурсного планування всього проекту в цілому. Для вирішення цієї проблеми необхідно ввести тимчасові обмеження на кожну із стадій життєвого циклу. Перехід здійснюється відповідно до плану, навіть якщо не вся запланована робота виконана. План складається на основі

статистичних даних, отриманих у попередніх проектах та особистого досвіду розробників.

3. Огляд основних гнучких методологій розробки

Гнучка модель (Agile model)

Являє собою сукупність різних підходів до розробки ПЗ. Включає серії підходів до розробки програмного забезпечення, орієнтованого на використання ітеративної розробки (в Scrum ітерації називаються спринтами), динамічне формування вимог і забезпечення їхньої 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 і з замовником напряму. Мається на увазі, що замовник взаємодіє з командою, команда з замовником - усі між собою. Це дозволяє обмінюватися досвідом між учасниками команди і клієнтом і кожному з них впливати на прийняття рішень. За рахунок такого підходу знижуються ризики втрати часу і грошей і підвищується здатність команди вирішувати складні нестандартні завдання з високим ступенем невизначеності.

Однак взаємодії всіх і з усіма можуть вилитися у хаос, що впливає на всі сфери розробки. Тому використовуючи Agile потрібно розуміти обмеження: команди повинні бути невеликі, учасники повинні бути компетентні та мотивовані, ітерації короткі з максимально зрозумілими цілями, встановлені чіткі обмеження за часом і кінцевий результат повинен бути очевидним.

Плюси:

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

Мінуси:

- велика кількість мітингів і обговорень, що може збільшити час розробки продукту;
- складно планувати процеси, так як вимоги постійно змінюються;
- рідко використовується для реалізації великих проєктів.

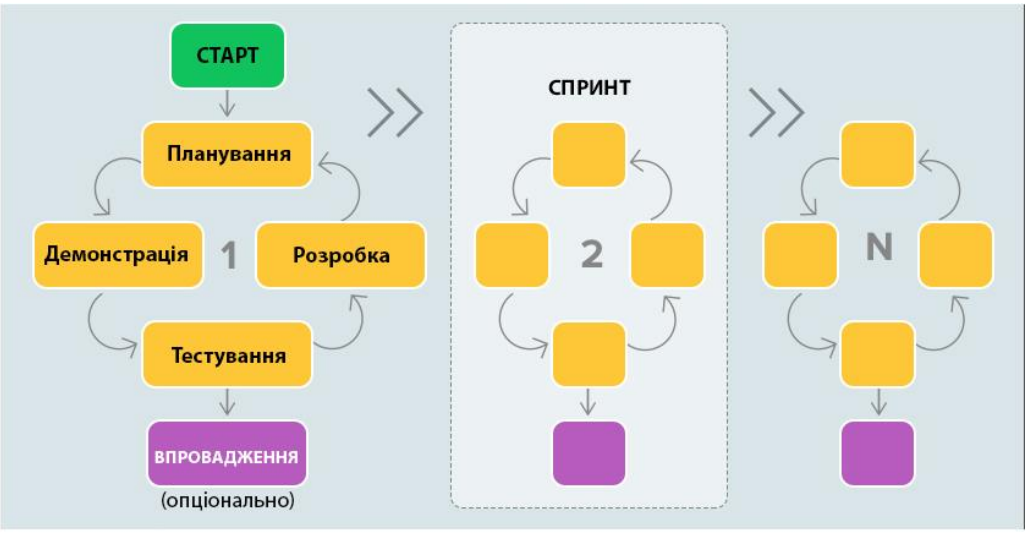

Рис.7. Agile model

Lean

Ідея підходу **Lean** полягає в тому, що ми ощадливо ставимося до ресурсів (у тому числі часу) і вирішуємо завдання найпростішим способом. Наприклад: ми не робимо весь продукт, щоб зрозуміти, що він нікому не потрібен – ми робимо лендінг і форму підписки і даємо на нього рекламу, щоб перевірити, що цей продукт викликає інтерес клієнтів і можна прийняти усвідомлене рішення про необхідність розробки.

Коли доходить до розробки продукту, або робиться якесь поліпшення, виробниче або інженерне, ми спочатку робимо його MVP (minimum viable product). Термін MVP зараз широко поширений і застосовується повсюдно, але він народився саме з Lean підходу. MVP - така версія продукту, що виконує свою головну функцію і при цьому її не відхиляють клієнти і визнають її корисність. Звичайно, її можна покращувати і покращувати, але загалом продукт на стадії MVP повинен бути корисним, зрозумілим клієнтові і таким, щоб можна було прийняти рішення про його подальших поліпшень або визнати експеримент невдалим і тестувати нову гіпотезу, витративши при цьому якомога менше ресурсів.

Загалом Lean підхід орієнтується на тестування потреб і цінностей і потрапляння в очікування ринку мінімальними засобами. Lean-підхід не про "технічні" проблеми і їхні рішення інженерними засобами, а про підприємницький підхід до вирішення завдань. Lean передбачає, що команда шукає найпростіше рішення для досягнення результату: технічно, організаційно і т.п., спрощуючи все, що не є дійсно важливим.

У спрощенні сила і головна "пастка" Lean: прагнення все спростити іноді призводить до ситуацій, в яких продукт спрощують настільки, що губляться дійсно важливі функції і продукт по суті виявляється непотрібним, оскільки не несе цінності користувачеві.

Scrum

Scrum – 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. Перед початком кожного спринту проводиться планування (Sprint Planning), на якому проводиться оцінка вмісту списку завдань із розвитку продукту (Product Backlog) і формування беклогу на спринт (Sprint Backlog), у рамках яких і діє команда. 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.

Плюси:

- 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.

Мінуси:

- 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".

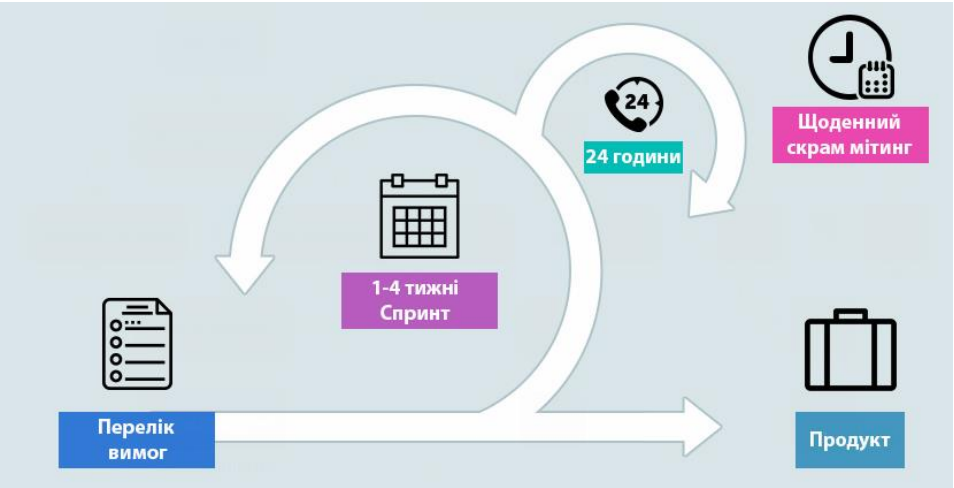

Рис. 8. Модель Scrum

За **Kanban** методологією проект ділиться на етапи, що візуалізуються у вигляді канбан-дошки. Етапи, це наприклад: планування, розробка, тестування, реліз і т.п. Завдання у вигляді "карток" переміщуються з етапу на етап. Нові завдання можна додавати у будь-який час. Завдання закривається не по закінченню конкретного часу, а по зміні статусу на "завершено". Kanban — методологія з концепції "ощадливого виробництва".

RUP (Rational Unified Process) – розробка продукту за даним методом складається з чотирьох фаз (початкова стадія, уточнення, побудова, впровадження), кожна з яких включає в себе одну або декілька ітерацій. RUP величезна методологія, котру важко описати одним абзацом тексту, але методи, рекомендовані RUP засновані на статистиці комерційно успішних проектів.

DSDM (Dynamic Systems Development Model) – методологія, що демонструє набір принципів, визначених типів ролей і технік.

Принципи спрямовані на головну мету - здати готовий проект вчасно і вкластися у бюджет, з можливістю регулювати вимоги під час розробки. DSDM належить до гнучких методологій розробки програмного забезпечення, а також розробок, що не входять у сферу інформаційних технологій.

RAD (Rapid Application Development) – методологія швидкої розробки додатків, що передбачає застосування інструментальних засобів візуального моделювання (прототипування) і розробки. RAD передбачає невеликі команди розробки, терміни до 4 місяців й активне залучення замовника з ранніх етапів. Дана методологія спирається на вимоги, але також існує можливість їхніх змін під час розробки системи. Такий підхід дозволяє скоротити витрати і звести час розробки до мінімуму.

XP (Extreme Programming) – методологія, орієнтована на постійну зміну вимог до продукт, пропонує 12 підходів для досягнення ефективних результатів у подібних умовах. Серед них:

- Швидкий план і його постійна зміна
- Простий дизайн архітектури;
- Часте тестування;
- Участь одночасно двох розробників в одному завданні або навіть за одним робочим місцем;
- Постійна інтеграція і часті невеликі релізи.

Існує безліч варіантів моделей розробки ПЗ. Вибір того чи іншого варіанту залежить від особливостей і вимог проєкту, моделей оплати. Не завжди вподобана з опису модель буде найкращою для реалізації саме вашого проекту. Частково методології перетинаються і схожі одна на одну, але тим не менш, кожна знаходить своїх шанувальників.