**ЛЕКЦІЯ-5. Методології розробки програмного забезпечення**

1. **Методології розробки програмного забезпечення**

Будь-яка теоретична або практична сфера діяльності використовує властиві тільки їй способи вирішення поставлених завдань. Ці способи називаються методами. Метод - це спосіб досягнення якої-небудь мети, рішення конкретної задачі; сукупність прийомів або операцій практичного або теоретичного освоєння дійсності.

**Методологія** *–* ***сукупність методів, що застосовуються в якій-небудь області людської діяльності****.*

Методологія — це, по-перше — систематизована сукупність підходів, способів, методів, прийомів та процедур, що застосовуються в процесі наукового пізнання та практичної діяльності для досягнення наперед визначеної мети. Такою метою в науковому пізнанні є отримання істинного знання або побудова наукової теорії та її логічного обґрунтування, досягнення певного ефекту в експерименті чи спостереженні тощо.

*Методологія - принципи і способи організації діяльності робочої группи для створення програмного продукту*

*Методологія розробки ПЗ полягає в організації процесу побудови ПЗ і забезпечення управління цим процесом для того, щоб гарантувати виконання вимог як до самої системи, так і до характеристик процесу розробки.*

Вибір методології розробки ПЗ має ключове значення для подальшого успіху проекту. Щоб повною мірою задовольнити всі вимоги замовника, компаніям, після детального аналізу, необхідно підібрати оптимальну методологію розробки для кожного конкретного випадку.

В залежності від поставленого завдання, необхідно визначити сучасні технології розробки, застосування яких буде найбільш доцільним і ефективним.

Не існує якоїсь однієї, єдино правильної методології, оптимальної для будь-якого проекту. У кожному конкретному випадку **правильний вибір методології розробки залежить від ряду чинників:**

* Масштабу проекту;
* Критичності проекту;
* Кількості і розподілу повноважень учасників проекту;
* Ступеня новизни проекту;
* Очікуваної тривалості проекту;
* Вимог Замовника

**Основними завданнями, рішення яких повинна забезпечувати методологія розробки ПЗ, є наступні:**

* ***забезпечення створення*** ПЗ, що відповідає цілям і завданням підприємства і відповідним пропонованим до них вимогам;
* ***гарантія створення*** ПЗ з заданими параметрами протягом заданого часу в рамках обумовленого заздалегідь бюджету;
* ***простота супроводу***, модифікації та розширення ПЗ з метою забезпечення його відповідності умовам, що змінюються при роботі підприємства;
* з***абезпечення створення*** ПЗ, що відповідає вимогам відкритості, переносимості та масштабованості;
* ***можливість використання*** в створюваному ПЗ розроблених раніше: ПЗ, БД, засобів обчислювальної техніки, телекомунікацій.

На сьогоднішній день існує не так багато методологій, особливо повних, тобто враховуючих всі стадії життєвого циклу програмного забезпечення. Саме методологія визначає, які мови та системи будуть застосовуватися для розробки ПЗ і, багато в чому, рекомендує, який технологічний підхід буде при цьому використано.

Зазвичай, вибір між традиційними і гнучкими (agile) методологіями значною мірою визначається бажаним ступенем формалізації процесу розробки. Для великих, розрахованих на тривалий термін проектів, як правило, рекомендуються більш адекватні методології з більшим ступенем формалізації, в той час як для менш масштабних проектів, розрахованих на оперативну реалізацію, підходять менше формалізовані гнучкі методи розробки.

***Rational Unified Process***

*Rational Unified Process –* (RUP)*— методологія розробки програмного забезпечення, створена компанією Rational Software. Уніфікований процес дозволяє створювати складні програмні системи.*

*RUP - суворий до розподілу завдань і відповідальності всередині організації-розробника. Його призначення – гарантувати створення точно в термін і в рамках встановленого бюджету якісного ПЗ, відповідно потребам кінцевих користувачів.* RUP сприяє підвищенню продуктивності колективної розробки і надає найкраще з накопиченого досвіду по створенню ПЗ, «посредством» посібників, шаблонів і настанов по користуванню інструментальними засобами для всіх критично важливих робіт, протягом ЖЦ створення і супроводу ПЗ. Надаючи кожному члену групи доступ до тієї ж самої базі знань, незалежно від того, чи розробляє він вимоги, проектує, виконує тестування або управляє проектом - RUP гарантує, що всі члени групи використовують спільну мову моделювання, процес, мають узгоджене бачення того, як створювати ПЗ. В якості мови моделювання в загальній базі знань використовується Unified Modeling Language (UML), що є міжнародним стандартом.

Особливістю RUP є те, що в результаті роботи над проектом створюються і удосконалюються моделі. Замість створення величезної кількості паперових документів, RUP спирається на розробку і розвиток семантично збагачених моделей, що всебічно представляють розроблювану систему. RUP - це керівництво по тому, як ефективно використовувати UML. Стандартна мова моделювання, що використовується усіма членами групи, робить зрозумілими для всіх описи вимог, проектування та архітектуру системи.

*RUP підтримується інструментальними засобами (Rational Requisite Pro, Rational Rose 98, Rational Purify), які автоматизують великі розділи процесу. Вони використовуються для створення і вдосконалення різних проміжних продуктів на різних етапах процесу створення ПЗ, наприклад, при візуальному моделюванні, програмуванні, тестуванні і т.д.*

**В основі методології лежать 6 основних принципів:**

* компонентна архітектура, реалізована і тестована на ранніх стадіях проекту;
* робота над проектом в згуртованої команді, ключова роль у якій належить архітекторам;
* рання ідентифікація та безперервне усунення можливих ризиків;
* концентрація на виконанні вимог замовників до виконуваній програмі;
* очікування змін у вимогах, проектних рішеннях і реалізації в процесі розробки;
* постійне забезпечення якості на всіх етапах розробки проекту.

RUP дозволяє приділити максимум уваги усуненню ризиків і забезпечити надійний контроль процесу розробки; орієнтований на ітераційний підхід в ході розробки і максимально точне виконання вимог Замовника.

RUP використовує ітеративну модель розробки. В кінці кожної ітерації (в ідеалі триваючої від 2 до 6 тижнів) проектна команда повинна досягти запланованих на дану ітерацію цілей, створити або допрацювати проектні артефакти і отримати проміжну, але функціональну версію кінцевого продукту. Ітеративна розробка дозволяє швидко реагувати на мінливі вимоги, виявляти й усувати ризики на ранніх стадіях проекту, а також ефективно контролювати якість створюваного продукту.

1. **Ітеративна модель. Базові поняття ітерації**

Не всі моделі життєвого циклу послідовні. Існують також ітеративні (або інкрементальні) моделі, в яких використовується інший підхід. Замість однієї тривалої послідовності дій тут весь життєвий цикл продукту розбитий на ряд окремих міні-циклів. Причому кожен з них складається з тих самих базових стадій моделі життєвого циклу. Ці міні-цикли називаються ітераціями.

У кожній з ітерацій відбувається розробка окремого компонента системи, після чого цей компонент додається до вже раніше розробленим функціоналу.

Ітеративна модель не передбачає повного обсягу вимог для початку робіт над продуктом. Розробка програми може починатися з вимог до частини функціоналу, які можуть згодом доповнюватися і змінюватися. Процес повторюється, забезпечуючи створення нової версії продукту для кожного циклу.

В дещо спрощеному вигляді, ітеративна модель складається з чотирьох основних стадій, які повторюються в кожній з ітерацій (plan-do-check-act):

* визначення і аналіз вимог;
* дизайн і проектування - відповідно до вимог. Причому дизайн може як розроблятися окремо для даної функціональності, так і доповнювати вже існуючий;
* розробка і тестування - кодування, інтеграція і тестування нового компонента;
* фаза рев'ю - оцінка, перегляд поточних вимог і пропозиції доповнень до них.

За результатами кожної ітерації приймається рішення - чи будуть використані її результати для доповнення існуючої функціональності в якості вхідної точки для початку наступної ітерації (т.зв. інкрементальне прототипування). В кінцевому підсумку, досягається точка, в якій всі вимоги були втілені в продукті - відбувається реліз.

У математичних термінах, ітеративна модель являє реалізацію методики послідовної апроксимації - тобто, поступове наближення до образу готового продукту: ключ до успішного використання цієї моделі - сувора валідація вимог і ретельна верифікація розробляється функціонально в кожній з ітерацій.

Основні стадії процесу розробки в ітеративній моделі фактично повторюють модель водоспаду. У кожній ітерації створюється програмне забезпечення, яке потребує тестування на всіх рівнях.

Плюси і мінуси ітеративної моделі:

+ - раннє створення працюючого ПО;

+ - гнучкість - готовність до зміни вимог на будь-якому етапі розробки;

+ - кожна ітерація - маленький етап, для якого тестування і аналіз ризиків забезпечити простіше, ніж для всього життєвого циклу продукту.

- - кожна фаза - самостійна, окремі ітерації не накладає;

- - можуть виникнути проблеми з реалізацією загальної архітектури системи, оскільки не всі вимоги відомі до початку проектування.

Коли використовувати ітеративну модель:

- для великих проектів;

- коли відомі, принаймні, ключові вимоги;

- коли вимоги до проекту можуть змінюватися в процесі розробки.

Ітеративна модель є ключовим елементом так званих «гнучких» (Agile) підходів до розробки програмного забезпечення.

#

Повний ЖЦ (життєвий цикл) розробки продукту складається з чотирьох фаз, кожна з яких включає в себе одну або декілька ітерацій



* 1. ***Початкова стадія (Inception)***

У фазі початковій стадії:

* формуються бачення і межі проекту;
* створюється економічне обґрунтування (business case).
* визначаються основні вимоги, обмеження і ключова функціональність продукту.
* створюється базова версія моделі прецедентів.
* оцінюються ризики.

При завершенні початкової фази оцінюється досягнення віхи цілей життєвого циклу (англ. Lifecycle Objective Milestone), яке припускає угоду зацікавлених сторін про продовження проекту.

* 1. ***Уточнення (Elaboration)***

У фазі «Уточнення» виробляється аналіз предметної області та побудова виконуваної архітектури. Це включає в себе:
    Документування вимог (включаючи детальний опис для більшості прецедентів).
    Спроектовану, реалізовану і відтестувати виконувану архітектуру.
    Оновлене економічне обґрунтування і більш точні оцінки термінів і вартості.
    Знижені основні ризики.

Успішне виконання фази розробки означає досягнення віхи архітектури життєвого циклу.

* 1. ***Конструювання (Construction)***

На етапі конструювання відбувається реалізація більшої частини функціональності продукту. Фаза Конструювання завершується першим зовнішнім релізом системи.

* 1. ***Впровадження (Transition)***

У фазі «Впровадження» створюється фінальна версія продукту і передається від розробника до замовника. Це включає в себе програму бета-тестування, навчання користувачів, а також визначення якості продукту. У випадку, якщо якість не відповідає очікуванням користувачів або критеріям, встановленим у фазі Початок, фаза Впровадження повторюється знову. Виконання всіх цілей означає досягнення віхи готового продукту (Product Release) та завершення повного циклу розробки.

***Відмінні риси RUP:***

   RUP - це ітеративний процес (Controlled Interactive);
    Передбачає наскрізне застосування апарату Use Cases (Use Case Driven);
    Особлива увага приділяється розробці архітектури;

    Включає управління вимогами та змінами;

    Базується на візуальному моделюванні.

***Microsoft Solution Framework***

*Microsoft Solution Framework -* методологія розробки ПЗ, що запропонована корпорацією Microsoft. MSF спирається на практический досвід Microsoft і описує управління людьми і робочими процесами в процесі розробки .

**Головною метою MSF**, як і будь методології проектування додатків, є створення робочого додатки вчасно і в рамках встановленого бюджету. MSF пропонує добре зарекомендували себе практики планування, розробки та впровадження інформаційних технологій. У той же час MSF не є простим набором інструкцій, яким належить слідувати беззастережно, - **цей процес досить гнучкий і розширюємо.**

**Застосування MSF дає:**

* прискорення циклів розробки
* скорочення сукупної вартості
* більш ефективне виконання планових заходів
* оперативне реагування на незаплановані події
* створення масштабованих і надійних технологій
* підвищення ефективності використання інформаційних технологій

**MSF будується** на ряді моделей:

1. Team Model - модель групи. Описує колектив, де робота одного співробітника залежить від іншого.
2. Process Model - модель процесу. Дозволяє визначити принципи планування і контролю проектів.
3. Application Model - модель програми. Допомагає створювати застосування максимально використовуючи готові компоненти.
4. Enterprise Architecture Model - забезпечує прийняття рішення за технологіями і дуже важлива для ефективного використання нових технологій.
5. Solution Design Model - модель структури рішення. Показує яким має бути програмне застосування з погляду користувача.
6. Infrastructure Model - визначає принципи управління користувачами у великих мережах.
7. Total Cost of Ownership Model - дозволяє оцінювати витрати на інформаційні технології.

**В основі MSF** лежать принципи і образи мислення, що представляють собою роки досвіду. Орієнтація на них допоможе групам ефективно створювати якісні продукти.

**Наступні принципи** і концепції MSF направляють проектну групу на реалізацію якісного рішення. Кожен член групи повинен розуміти і приймати ці принципи при взаємодії з іншими учасниками групи, їх організацією та зацікавленими особами.

В основі MSF лежать дев'ять базових принципів:

* ***відкрита взаємодія.*** Щоб ваша група працювала ефективно, ви і інші учасники повинні ділитися відповідними рівнями інформації з членами групами і співробітниками підприємства. Група повинна розуміти природу створюваного рішення і те, як будуть взаємодіяти члени групи і зовнішні контакти. Складність полягає у визначенні відповідного рівня всіх взаємин та інформації, яка буде загальнодоступною;
* ***вироблення загального уявлення.*** Загальне уявлення підвищує ефективність і динамічність групи, що дозволяє учасникам швидко приймати виважені рішення в контексті реалізації поставленого подання. Загальне уявлення також допомагає заповнювати прогалини у вимогах при їх виявленні;
* ***наділення членів команди необхідними повноваженнями***. Надання повноважень членам групи - це не тільки один із способів вижити в постійно мінливому середовищі, а й спосіб творчого пошуку для успішного вирішення завдань і допомоги один одному. Якщо членам групи не дозволити працювати на максимумі, не тільки їх творчий потенціал буде звужений, їх моральний дух не буде на висоті і ви не зможете створити продуктивну групу;
* ***встановлення чітких меж повноважень та спільної відповідальності***. Наділені повноваженнями члени групи частіше відчувають себе більш відповідальними за свої рішення і за проект в цілому. Велика ступінь відповідальності членів групи веде до більш високій якості. Наприклад, якщо член групи заявляє, що завдання виконане, але виявляється, що рівень якості не відповідає необхідному, цей член групи відповідає за виправлення результатів. Заохочуючи позитивні результати і відповідальність, а не караючи за такі помилки, можна домогтися того, що члени групи будуть разом нести відповідальність за спільне рішення. Це мотивує найсильніших учасників допомагати іншим працювати максимально ефективно;
* ***реалізація інкрементних результатів***. Реалізація інкрементних результатів пов'язана з двома аспектами: *необхідно переконатися, що надаються результати оптимальні для зацікавлених осіб*; *необхідно визначити оптимальні інкременти для надання результатів, т. е. "частот доставки";*
* ***гнучкість, очікування змін і адаптування до них.*** Так як зміни можуть відбуватися часто і в найгірший час, наявність гнучкого способу для обробки змін допоможе вам мінімізувати простої, пов'язані з ними. Гнучкість означає, що організація готова до змін і може легко адаптуватися до нихі;
* ***нвестиції в якість***. Багато організацій використовують поняття якості (часто воно визначено дуже розмито), але не розуміють, як квантифікувати його. Якість - це те, що необхідно проактивно впроваджувати в життєвий цикл доставки рішення. Саме по собі воно не з'явиться;
* ***навчання на будь-якому досвіді***. Якщо всі рівні організації не будуть вчитися на тому, що раніше працювало і не працювало, як від них можна чекати поліпшень в майбутньому? Члени групи повинні розуміти, що навчання відбувається на всіх рівнях: *на рівні проекту, наприклад, при уточненні процесу для всього проекту; на рівні окремої особи, наприклад, для поліпшення взаємодії з іншими членами групи; на рівні організації, наприклад, при коригуванні показників якості, які збираються для кожного проекту;*
* ***співпраця з внутрішніми і зовнішніми клієнтами.*** Якщо клієнт працює разом з проектною групою, ви підвищуєте ймовірність успіху проекту. Але це не означає, що клієнти повинні робити всю роботу за вашу групу. Однак коли клієнти тісно і послідовно співпрацюють з проектною групою, отримане рішення більше відповідає їх вимогам. Співпраця з клієнтами взаємовигідно, тому що воно допомагає усунути невизначеність, скоротити час усунення проблем з вимогами та покращити розуміння групою ціннісного пропозиції вирішення за рахунок регулярних контактів.

**Базові принципи**, описані вище, визначають, як зорієнтувати групу, щоб домогтися успіху. Орієнтація окремих членів групи для більш ефективної роботи називається "способом мислення". Кожен образ мислення допомагає членам групи домогтися своєї мети при реалізації рішення. В ідеалі члени групи так комфортно себе почувають в рамках цих образів мислення, що вони використовують їх не тільки на роботі, але і поза нею.

**Образи мислення яким має слідувати кожен член групи:**

1. Сприяння формуванню групи рівноцінних колег. Якщо ваша організація може застосувати базові принципи MSF, зокрема що стосуються стимулюючих повноважень і відповідальності, чи має сенс працювати над проектом з ієрархічною структурою? Якщо всі розуміють головну і допоміжні цілі, поділяють загальне уявлення, знають свої ролі і відповідальності, то всі діють як рівні і ставитися до них слід однаково. Це не слід розцінювати як пропозицію організувати анархію або комітет. Це означає, що всі члени групи розділяють відповідальність за успішну реалізацію проекту. Кожна роль відповідальна за відповідні аспекти проекту, а всі ролі несуть колективну відповідальність за проект в цілому. Як видно, роль керівника програми як і раніше є, але вона сконцентрована на реалізації проекту в рамках заданих обмежень, а не на управлінні членами групи.
2. Фокусування на цінності для бізнесу. Успіх вимірюється з точки зору цінності для бізнесу. Це означає надання не тільки того, що клієнтам необхідно, але й того, чого вони хочуть і цінують. Для цього кожен член групи повинен розуміти, що саме клієнти вважають цінним. Якщо клієнт не отримає бізнес-цінність, проект піддається ризику, наприклад, група може збитися зі шляху, витратити багато зайвого часу, зусиль і коштів, або ж проект може бути взагалі скасований.
3. Рішення має бути перспективним. При аналізі компонентів рішення через розмір та складність більшості проектів члени групи іноді занадто придивляються до маленьких деталей і забувають про рішення в цілому. Тому стільки уваги приділяється спільним поданням. У міру того як члени групи надають свої компоненти, їм необхідно поглянути на загальну мету і подання рішення. Занадто часто підгрупа оптимізує свою область, вважаючи, що вони діють заради загального блага. Але потім вони виявляють, що їм необхідно переробити певні аспекти, щоб їх компонент був узгоджений з рішенням. Члени групи заплутуються в деталях і забувають про цілісне рішення.
4. Відношення до своєї роботи з повагою. Група не тільки повинна інвестувати в якість, але учасники також повинні бачити, що якість - така ж їх відповідальність, як і інших членів групи, тому свої завдання не слід делегувати комусь іншому. Якість - це спільна відповідальність в рамках усього життєвого циклу розробки рішення. Цей спосіб мислення не тільки дозволяє підвищити якість власних результатів члена групи, але і покращує ефективність процесів і управління проектом. Завдяки йому кожен член групи прагне краще зрозуміти, які навички необхідні для досягнення спільної мети. Відстежуючи якість власної роботи та надаючи найкращі результати, члени групи зможуть постійно покращувати кінцевий продукт.
5. Постійне навчання. Іноді для досягнення кінцевої мети недостатньо пишатися своїми навичками. Члени групи повинні опановувати новими навичками, щоб стати краще. Враховуючи, що більшість проектів, груп і середовищ унікальні, кожен проект надає можливості для навчання, експерименту, а також вдосконалення навичок, процесів і процедур. Щоб скористатися цими можливостями, безперервні процеси навчання та адаптації повинні існувати на всіх рівнях організації, а не тільки для членів групи.
6. Впровадження рівнів якості обслуговування. Якість обслуговування (QoS) визначає операційні характеристики рішення, наприклад очікуваний рівень доступності. Важливо, щоб зацікавлені особи і члени групи, не тільки фахівці з архітектури, розуміли принципи якості обслуговування і знали, як їх реалізація впливає на кінцевий результат. В іншому випадку зацікавлені особи і члени групи, ймовірно, будуть робити неявні припущення про те, як рішення повинно працювати. Так як подібні припущення рідко узгоджені, кожному члену групи необхідно приймати явні рішення з самого початку роботи над проектом, щоб забезпечити потрібні рівні якості обслуговування. Таким чином, неявні припущення перетворюються в явні вимоги до QoS, які явно вбудовуються в рішення з самого початку роботи, а не надбудовуються згодом.
7. Заохочення солідарності. З точки зору розробки ПЗ солідарність означає довіру, чесність, відповідальність і повага для всіх аспектів роботи. Вона, зокрема, визначає:

       як взаємодіяти з членами групи, з організацією та зацікавленими особами;
        як брати участь у проекті та допомагати у реалізації проекту, в тому числі ефективно використовувати корпоративні, проектні та обчислювальні ресурси (до яких відносяться загальні ресурси, інформація та знання). Солідарні члени групи діють для загального блага.

1. Виконання свої зобов'язань. Незважаючи на багато вбудовані перевірки MSF покладається на довіру і повноправні повноваження, щоб домогтися яких, члени групи, в тому числі, повинні виконувати свої зобов'язання. MSF формує середовище, в якому члени групи і зацікавлені особи можуть довіряти результатам своїх колег. Так як проект - це набір незалежних дій, то якщо один член групи не виконує свої зобов'язання, він ставить під загрозу весь проект.

**Модель управління**

Модель управління (раніше - модель процесів) призначена надання потрібних інструкцій потрібним людям в потрібний час.

Модель структурована, щоб допомогти групі швидко приходити до консенсусу про те, як реалізувати різні аспекти вирішення.

Модель управління - це гнучкий компонент MSF, який успішно використовується для поліпшення управління проектом, зниження ризиків, підвищення якості рішення і прискорення розробки.

Так як MSF повністю налаштовується, передбачається, що організація адаптує модель управління у відповідності зі своїми бізнес-процесами та підходами до розробки рішень.

Модель управління MSF об'єднує управління проектом із впровадженням необхідних процесів. Управління проектом фокусується на оптимізації процесу розробки рішення та ефективному використанні ресурсів проекту. Впровадження процесів фокусується на визначенні, побудові і розгортанні рішення, яке відповідає потребам і очікуванням зацікавлених осіб.

**Модель управління** MSF дозволяє реалізувати управління проектом та впровадження процесів за допомогою пересічних напрямків . Модель управління MSF складається з п'яти пересічних напрямів:

* *ідея;*
* *планування;*
* *збірка;*
* *стабілізація;*
* *розгортання*

Модель процесу SDD являє собою один з варіантів спіральної моделі: коли один цикл впровадження близький до завершення, вже повинен плануватися наступний

Цикл (виток спіралі) розробки включає чотири фази і завершується випуском версії продукту. Кожна фаза являє собою певну послідовність дій і завершується віхою (milestone):

• **Перша фаза** - Аналіз (Envisioning). На даному етапі формується уявлення про продукт на даному витку спіралі. Це гарантує, що розробляється продукт буде відповідати як поточним, так і перспективним цілям компанії, а також допоможе скоректувати напрямок розвитку компанії. Дана стадія вимагає глибокого осмислення цілей проекту. Формування уявлення дозволяє уникнути, наприклад, інвестування в незначні або неефективні проекти. В результаті цієї стадії складається уявлення про продукт, визначаються його функціональні можливості і оцінюються результати.

• **Друга фаза** - Планування (Planning). З точки зору Microsoft, планування - це процес узгодження вимог споживачів і групи проекту, що стосуються кінцевого продукту та напрямки розробки продукту. Це метод прогнозування ризиків, вироблення пріоритетів та оцінки графіка робіт і необхідних ресурсів. Фаза планування завершується схваленням плану проекту, який включає функціональну специфікацію - комбіновані плани для кожного члена групи відповідно до вимог моделі команди і графік робіт. Функціональна специфікація достатньо деталізована, щоб дозволити групі проекту визначити потреби в ресурсах і її зобов'язання. Віхою даної фази є затвердження плану проекту.
• **Третя фаз**а - Розробка (Developing). Стадія розробки завершується реалізацією можливостей продукту і перевіркою їх на практиці. Група розробки визначає проміжні етапи випуску продукту, кожен з яких включає повний цикл тестування, налагодження і внесення виправлень. На цьому етапі споживачі і група розробки оцінюють функціональні можливості продукту і перевіряють оптимальність планів розгортання та підтримки. Розробка в цілому завершується, а всі нереалізовані можливості документуються для включення в наступні версії. Віхою даної фази є завершення розробки, альфа-версія (передається тестерам, користувачам, починається усунення помилок).

• **Четверта фаза** - Стабілізація (Stabilizing). На цій стадії акцент переноситься з розробки рішення на перевірку його працездатності в реальних умовах і на повномасштабне тестування. Стадія стабілізації завершується випуском продукту, який передається групам розгортання та супроводу. Групі проекту доручають створення наступної версії або підключають до роботи над іншими проектами. Віхою даної фази є реліз продукту.

Контрольними точками процесу є віхи (milestones). Робота команди орієнтована на досягнення віх, що супроводжується появою і фіксацією того чи іншого відчужуваного матеріалу (документа, програми, протоколу і т.д.). Віха - це час проведення інспекцій (фазових оглядів), на яких обговорюються досягнуті результати і приймаються рішення.

Розгортання – просте розгортання і перехід в експлуатацію. Забезпечення виконання потреб і очікувань ІТ- і бізнес-підрозділів. (Управління релізами, інфраструктура доставки, управління збірками, адміністрування)