**Тема 2. Життєвий цикл і етапи розробки програмного забезпечення**

У процесі створення ПЗ можна виділити 4 базових етапи / стадії (рис.3.1):

− Специфікація - визначення основних вимог.

− Розробка – створення ПЗ у відповідності зі специфікаціями.

− Тестування - перевірка ПЗ на відповідність вимогам клієнта.

− Супровід / Модернізація – розвиток ПЗ відповідно до змін потреб замовника.



Рисунок 3.1 – Схема життя програмного забезпечення

 **Процеси життєвого циклу програмного забезпечення**

 *Життєвим циклом програмного забезпечення називають* період від моменту появи ідеї створення деякого програмного забезпечення до моменту завершення його підтримки фірмою-розробником або фірмою, яка виконувала супровід. Життєвий цикл програмного забезпечення – період часу, що починається з моменту прийняття рішення про необхідність створення програмного продукту і закінчується в момент його повного вилучення з експлуатації.

 Склад процесів життєвого циклу регламентується міжнародними стандартами

 **ISO / ІEC 12207: 1995** *– «Information Technologe - Software Life Cycle Processes»* («Інформаційні технології – процеси життєвого циклу програмного забезпечення »).

 **ISO -** *International Organization for Standardization* **–** Міжнародна організація по стандартизації.

 **I**ЕС **–** *International Electrotechnical Com* Mission – Міжнародна комісія з електротехніки. Цей стандарт описує структуру життєвого циклу програмного забезпечення та його процеси.

 *Процес життєвого циклу* визначається як сукупність взаємопов'язаних дій, які перетворюють деякі вхідні дані у вихідні.

 На рисунку 3.2 представлені процеси життєвого циклу за вказаним стандартом. Кожен процес характеризується певними завданнями і методами їх вирішення, а також вихідними даними і результатами.



 Перехід від ручних засобів розробки ПЗ до промислового виробництва програм потребував розвитку теоретичних основ розробки ПЗ. Постійна необхідність внесення змін до програми викликана як помилками, так і розвитком вимог до них, є принциповою властивістю програмного забезпечення.

 Діяльність, пов'язана з вирішенням широкого ряду завдань для постійного розвитку, отримала назв*у супроводу програмного забезпечення*.

 Якщо зусилля, що спрямовані на модернізацію ПЗ, перевищують вигоду від його використання, говорять про *моральне старіння програм*.

 Оскільки розробка і супровід ПЗ фактично є проектною діяльністю, частина ключових понять управління проектів знайшла широке застосування в *програмній інженерії*. Таким і поняття життєвого циклу проекту (Project Lifecycle Management, PLM), що в програмной інженерії трансформувалося в поняття життєвого циклу програмного забезпечення.

Модель ЖЦ є абстракцією реального процесу, в якій відсутні деталі, несуттєві з точки зору призначення моделі. Поняття ЖЦ виникло під впливом потреби в систематизації робіт у процесі розробки ПЗ. Систематизація була першим етапом на шляху до автоматизації процесу розробки ПЗ.

 Наступними кроками переходу до автоматизації процесу розробки ПЗ були такі: встановлення технологічних маршрутів розробки ПЗ; визначення можливості їх автоматизації та виявлення ризиків; розробка інструментів для автоматизації.

 Спочатку з'явилися *інструменти підтримки розробки програмного коду і налагодження програм (60-і рр)*. Після усвідомлення недостатності таких засобів для суттєвого підвищення якості програм і створення інструментів управління процесом розробки виникло поняття життєвого циклу ПЗ. Виявлення закономірностей розвитку програмного забезпечення відразу показало нерозвиненість методик конструювання ПЗ і недостатність тестування для визначення якості програмних продуктів. Також на цьому етапі стало зрозуміло, що нечіткість завдання на створення ПЗ викликає більшість проблем розробки та перевірки програм.

 У результаті виникли вимоги до постановки задачі, сформувалися підходи до управління вимогами на етапі аналізу та інструменти зв'язку вимог на етапах аналізу та реалізації.

 Використання поняття життєвого циклу дозволяє вибрати найбільш ефективні підходи для задач певного етапу життя ПЗ. Залежно від особливостей процесів розробки і супроводу програм існують різні моделі ЖЦ.

 Використання певної моделі ЖЦ дозволяє визначитися з основними моментами процесу замовлення, розробки і супроводу ПЗ навіть недосвідченому програмісту. Також використання моделей дозволяє чітко зрозуміти, в який період переходити від версії до версії, та які дії з удосконалення виконувати і на якому етапі.

 Знання про закономірності розвитку програмного продукту, які відображаються в обраній моделі ЖЦ, дозволяють отримати надійні орієнтири для планування процесу розробки і супроводу ПЗ, економно витрачати ресурси і підвищувати якість управління всіма процесами.

 **Процес розробки програмного забезпечення**

 *Процес розробки (development process)* відповідно до стандарту передбачає дії і завдання, що виконуються розробником, і охоплює роботи зі створення програмного забезпечення і його компонентів відповідно до заданих вимог, включаючи оформлення проектної та експлуатаційної документації, а також підготовку матеріалів, необхідних для перевірки працездатності і відповідності якості програмних продуктів, матеріалів, необхідних для навчання персоналу, і т.д.

 *За стандартом процес розробки включає такі дії:*

* *підготовчу роботу –* вибір моделі життєвого циклу (див. далі), стандартів, методів і засобів розробки, а також складання плану робіт;
* *аналіз вимоги до системи –* визначення її функціональних можливостей, користувальницьких вимог, вимог до надійності і безпеки, вимог до зовнішніх інтерфейсів і т. д;
* *проектування архітектури системи –* визначення складу необхідного обладнання, програмного забезпечення та операцій, які виконуються обслуговуючим персоналом;
* *аналіз вимог до програмного забезпечення –* визначення функціональних можливостей, включаючи характеристики продуктивності, середовища функціонування компонентів, зовнішніх інтерфейсів, специфікацій надійності та безпеки, ергономічних вимог, вимог до даних, що використовуються, установці, прийманні, користувацької документації, експлуатації та супроводу;
* *проектування архітектури програмного забезпечення –* визначення структури програмного забезпечення, документування інтерфейсів його компонентів, розробку попередньої версії документації користувача, а також вимог до тестів і плану інтеграції;
* *детальне проектування програмного забезпечення –* докладний опис компонентів програмного забезпечення та інтерфейсів між ними, оновлення користувальницької документації, розробка та документування вимог до тестів і плану тестування компонентів програмного забезпечення, оновлення плану інтеграції компонентів;
* *кодування і тестування програмного забезпечення –* розробка та документування кожного компонента, а також сукупності тестових процедур і даних для їх тестування, тестування компонентів, оновлення документації користувача, оновлення плану інтеграції програмного забезпечення;
* *інтеграція програмного забезпечення –* збірка програмних компонентів відповідно до плану інтеграції та тестування програмного забезпечення на відповідність кваліфікаційним вимогам, що представляють собою набір критеріїв або умов, які необхідно виконати, щоб кваліфікувати програмний продукт як відповідний своїм специфікаціям і готовий до використання в заданих умовах експлуатації;
* *кваліфікаційне тестування програмного забезпечення –* тестування програмного забезпечення в присутності замовника для демонстрації його відповідності вимогам і готовності до експлуатації; при цьому перевіряється також готовність і повнота технічної документації користувача;
* *інтеграція системи –* збірка всіх компонентів системи, включаючи програмне забезпечення та обладнання;
* *кваліфікаційне тестування системи –* тестування системи на відповідність вимогам до неї і перевірка оформлення та повноти документації;
* *установка програмного забезпечення –* установка програмного забезпечення на обладнанні замовника і перевірка його працездатності;
* *приймання програмного забезпечення –* оцінка результатів кваліфікаційного тестування програмного забезпечення і системи в цілому та документування результатів оцінки спільно із замовником, остаточна передача програмного забезпечення замовнику.

 Зазначені дії можна згрупувати, умовно виділивши такі *основні етапи розробки програмного забезпечення:*

1. Постановка задачі (стадія «Технічне завдання»);
2. Аналіз вимог і розробка специфікацій (стадія «Ескізний проект»);
3. Проектування (стадія «Технічний проект»);
4. Реалізація (стадія «Робочий проект»).
5. Підготовка і передача програми (стадія «Впровадження»)

 Традиційно розробка також включала етап *супроводу* (початку цього етапу відповідає стадія «Впровадження»).

 Умовність виділення етапів пов'язана з тим, що на будь-якому етапі можливе прийняття рішень, які вимагатимуть перегляду рішень, прийнятих раніше.

 **Стадія – Постановка задачі**

 У процесі *постановки завдання* чітко формулюють призначення програмного забезпечення і визначають основні вимоги до нього. Кожна вимога є описом необхідної або бажаної властивості програмного забезпечення.

 Розрізняють:

* *функціональні вимоги*, що визначають функції, які має виконувати програмне забезпечення;
* *експлуатаційні вимоги*, що визначають особливості його функціонування.

 Вимоги до програмного забезпечення, що має *прототипи*, зазвичай визначають за аналогію, враховуючи структуру і характеристики вже існуючого програмного забезпечення.

 Для формулювання вимог до програмного забезпечення, що не має аналогів, іноді необхідно провести спеціальні дослідження, що називаються передпроектними. У процесі таких досліджень визначають розрішення завдання, можливо, розробляють методи його рішення (якщо вони нові) і встановлюють найбільш суттєві характеристики програмного забезпечення, що розробляється. Для виконання передпроектних досліджень, як правило, укладають договір на виконання науково-дослідних робіт.

 У будь-якому випадку етап постановки завдання закінчується розробкою *технічного завдання*, що фіксує принципові вимоги, і прийняттям основних проектних рішень.

 **Стадія – Аналіз вимог і визначення специфікацій**

 *Специфікаціями* називають точний формалізований опис функцій і обмежень програмного забезпечення, що розробляється. Відповідно розрізняють функціональні і експлуатаційні специфікації.

 Сукупність специфікацій являє собою *загальну* логічну модель програмного забезпечення.

 Для отримання специфікацій виконують аналіз вимог технічного завдання, формулюють змістовну постановку задачі, вибирають математичний апарат формалізації, будують модель предметної області, визначають підзадачі і вибирають або розробляють методи їх вирішення. Частина специфікацій може бути визначена в процесі передпроектних досліджень і, відповідно, зафіксована в технічному завданні.

 На цьому етапі також доцільно сформувати тести для пошуку помилок в програмному забезпеченні, що проектується, та обов'язково вказати очікувані результати.

 **Стадія – Проектування**

 Основним завданням цього етапу є визначення *докладних* специфікацій програмного забезпечення.

 *Процес проектування складного програмного забезпечення зазвичай включає:*

* проектування загальної структури - визначення основних компонентів і їх взаємозв'язків;
* декомпозицію компонентів і побудову структурних ієрархій відповідно до рекомендацій блочно-ієрархічного підходу;
* проектування компонентів.

 Результатом проектування є *детальна модель* програмного забезпечення разом зі специфікаціями його компонентів всіх рівнів. Тип моделі залежить від обраного підходу (структурний, об'єктний або компонентний) і конкретної технології проектування. Однак у кожному разі процес проектування охоплює як проектування програм (підпрограм) і визначення взаємозв'язків між ними, так і проектування даних, з якими взаємодіють ці програми або підпрограми.

 *Прийнято розпізнавати також два аспекти проектування:*

* + *логічне проектування*, яке включає ті проектні операції, які безпосередньо не залежать від наявних технічних і програмних засобів, що складають середовище функціонування майбутнього програмного продукту;
	+ *фізичне проектування* – прив'язка до конкретних технічних і програмних засобів середовища функціонування, тобто врахування обмежень, що визначені у специфікаціях.

 **Стадія – Реалізація**

 Реалізація являє собою процес поетапного написання кодів програми обраною мовою програмування (кодування), їх тестування і налагодження.

 **Стадія – Супровід**

 Супровід – це процес створення і впровадження нових версій програмного продукту. Причинами випуску нових версій можуть служити:

* необхідність виправлення помилок, виявлених в процесі експлуатації попередніх версій;
* необхідність удосконалення попередніх версій, наприклад, поліпшення інтерфейсу, розширення складу виконуваних функцій або підвищення його продуктивності;
* зміна середовища функціонування, наприклад, поява нових технічних засобів та / або програмних продуктів, з якими взаємодіє програмне забезпечення, що супроводжується.

 На цьому етапі в програмний продукт вносять необхідні зміни, які так само як і в інших випадках, можуть зажадати перегляду проектних рішень, прийнятих на будь-якому попередньому етапі.

 Зі зміною моделі життєвого циклу програмного забезпечення роль цього етапу істотно зросла, так як продукти тепер створюються ітераційно: спочатку випускається порівняно проста версія, потім наступна з великими можливостями, потім наступна і т. д. Саме це і послужило причиною виділення етапу супроводу в окремий процес життєвого циклу відповідно до стандарту 1SO / IEC 12207.

 Розглянутий стандарт тільки називає і визначає процеси життєвого циклу програмного забезпечення, не конкретизуючи в деталях як реалізовувати або виконувати дії і завдання, що включені в ці процеси. Ці питання регламентуються відповідними методами, методиками і т. п. Перш ніж перейти до докладного розгляду останніх, проаналізуємо еволюцію схем розробки програмного забезпечення від моменту їх появи до теперішнього часу.

**Еволюція моделей життєвого циклу програмного забезпечення**

Моделі життєвого циклу є основою знань технологій програмування та інструментарію, які підтримує. У що технологія базується на певних уявленнях про життєвий цикл і організовує свої методи і інструменти навколо фаз і етапів ЖЦ.

Розвиток методологій програмування в 70-х рр XХ ст. привів до формування потреби вивчення життєвого циклу ПЗ. До цього часу моделі ЖЦ розвиваються і модифікуються, уточнюючи і доповнюючи дві базові моделі – каскадну і ітеративну. Ці зміни обумовлені необхідністю організаційної та технологічної підтримки проектів з розробки ПЗ.

Перехід від ручних засобів розроблення ПЗ до промислового виробництва програм потребував розвитку теоретичних основ розроблення ПЗ. Постійна необхідність внесення змін у програми як спричинена помилками, так і розвитком вимог до них, є принциповою властивістю програмного забезпечення. Діяльність, пов’язана з рішенням широкого ряду завдань для постійного розвитку, отримала назву супроводу програмного забезпечення. Якщо зусилля, спрямовані на модернізацію ПЗ, перевищують вигоду від його використання, говорять про моральне старіння програм.

Оскільки розроблення та супровід ПЗ фактично є проектною діяльністю, частина ключових понять управління проектів знайшла широке застосування у програмній інженерії. Таким є і поняття життєвого циклу проекту (Project Lifecycle Management, PLM), що в програмній інженерії трансформувалось у поняття життєвого циклу програмного забезпечення.

Протягом останніх тридцяти років у програмуванні змінилися три моделі життєвого циклу програмного забезпечення; каскадна (водоспадна), модель з проміжним контролем і спіральна.

 **Каскадна модель** **(waterflow model)**

*Водоспадна модель (каскадна схема)* запропонована в 1970 році Вінстоном Ройсом. і використовувалася в розробці програмного забезпечення (рис. 4.1), яке передбачало, що перехід на наступну стадію здійснюється після того, як повністю будуть завершені проектні операції попередньої стадії і отримані всі вихідні дані для наступної стадії.



Рисунок 4.1 – Каскадна схема розробки програмного забезпечення

Фактично вперше в процесі розробки ПЗ були виділені різні кроки розробки і розхитані примітивні уявлення про розробку ПЗ у вигляді аналізу системи та її кодування.

*Перевагами такої схеми є* :

* отримання в кінці кожної стадії закінченого набору проектної документації, що відповідає вимогам повноти і узгодженості;
* простота планування процесу розробки.

Саме таку схему і використовують зазвичай при блочно-ієрархічному підході до розробки складних технічних об'єктів, що забезпечує дуже високі параметри ефективності розробки.

Однак дана схема виявилася придатною тільки до створення систем, для яких в самому початку розробки вдавалося точно і повно сформулювати всі вимоги. Це зменшувало ймовірність виникнення в процесі розробки проблем, що пов'язані з прийняттям невдалого рішення на попередніх стадіях. На практиці такі розробки зустрічається вкрай рідко.

Каскадна модель ЖЦ ПЗ виникла для задоволення потреби в систематизації робіт ще на ранніх етапах розробки програм. Відповідно до цієї моделі програмні системи проходять у своєму розвитку дві фази: розробка, супровід.

Фази розбиваються на ряд етапів. Каскадна модель передбачає послідовне виконання всіх етапів проекту в суворо фіксованому порядку. Перехід на наступний етап означає повне завершення робіт на попередньому етапі (рис. 2.2.. Розробка починається з ідентифікації потреби в новому додатку, а закінчується передачею продукту розробки в експлуатацію. Всі етапи розробки програмного забезпечення регламентуються стандартами підприємства і державним з стандартам.

*Першим етапом фази розробки специфікація (Requirements Speсification) – розробка системних вимог, розробка вимог до ПЗ*.

На етапі постановки завдання замовник спільно з розробниками приймають рішення про створення системи. Визначення вимог включає опис загального контексту завдання, очікуваних функцій системи та її обмежень. У разі позитивного рішення починається аналіз системи відповідно до вимог. Розробники програмного забезпечення намагаються осмислити висунуті замовником вимоги і зафіксувати їх у вигляді специфікацій системи. *Призначення специфікацій – описувати зовнішню поведінку системи, а не її внутрішню організацію.* Перш ніж почати створювати проект за специфікаціями, вимоги мають бути ретельно перевірені на відповідність вихідним цілям, повноту, сумісність (несуперечливість) і однозначність**.**

*Завдання етапу аналізу* полягає в тому, щоб вибудувати опис програми у вигляді логічної системи, зрозумілої як для замовника та майбутніх користувачів, так і для виконавців проекту. На етапі специфікації обов'язково формується технічне завдання на розробку ПЗ.

Розробка проектних рішень, що відповідають на питання, як має бути реалізована система, щоб вона могла задовольняти певні вимоги, виконується на *етапі проектування (Design).* Оскільки складність системи в цілому може бути дуже великою, головним завданням цього етапу є послідовна декомпозиція системи до рівня очевидно реалізованих модулів або процедур. Результати виконання цього етапу оформляються як технічний проект, вимоги до документів якого встановлені стандартом*.*

На наступному *етапі реалізації (Construction), або кодування*, кожен з модулів програмується на найбільш придатній для даного застосування мові. З точки зору автоматизації цей етап традиційно є найбільш розвинений.

У каскадної моделі фаза розробки закінчується *етапом тестування (Testing and debugging),* автономного і комплексного, і передачею системи в експлуатацію (Installation).

*Фаза експлуатації (Використання) та супроводу* включає в себе всю діяльність щодо забезпечення нормального функціонування програмних систем, у тому числі фіксування розкритих під час виконання програм помилок, пошук їх причин та виправлення, підвищення експлуатаційних характеристик системи, адаптацію системи до навколишнього середовища, а також, при необхідності, і більш істотні роботи з удосконалення системи. Фактично відбувається еволюція системи.

У ряді випадків на цю фазу припадає більша частина коштів, які витрачаються в процесі життєвого циклу програмного забезпечення. Зрозуміло, що увага програмістів до тих чи інших етапів розробки залежить від конкретного проекту. Часто розробнику немає необхідності проходити через всі етапи, наприклад, якщо створюється невелика, добре зрозуміла програма з чітко поставленою метою.

*Коротка характеристика каскадної схеми розробки програмного забезпечення:*

− фіксований набір стадій;

− кожна стадія закінчується документованим результатом;

− наступна стадія починається тільки 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 і стабільні.

*Необхідність повернень на попередні стадії обумовлена такими причинами:*

* неточні специфікації, уточнення яких у процесі розробки може призвести до необхідності перегляду вже прийнятих рішень;
* зміна вимог замовника безпосередньо в процесі розробки;
* швидке моральне старіння технічних і програмних засобів, що використовуються;
* відсутність задовільних засобів опису розробки на стадіях постановки задачі, аналізу та проектування.

Відмова від уточнення (зміни) специфікацій призведе до того, що закінчений продукт не буде задовольняти потребам користувачів.

При відмові від врахування зміни обладнання і програмного середовища користувач отримає морально застарілий продукт.

Відмова від перегляду невдалих проектних рішень призводить до погіршення структури програмного продукту і, відповідно, ускладнить, розтягне за часом і здорожує процес його створення. Реальний процес розробки, таким чином, носить ітераційний характер.

**Ітеративна модель** **(Iterative and incremental development) –** модель з проміжним контролем.

Каскадна модель життєвого циклу не є ідеальною, оскільки лише дуже прості завдання проходять всі етапи без будь-яких ітерацій (повернень на попередні кроки процесу).

Наприклад, при програмуванні може виявитися, що реалізація деякої функції неефективна і вступає в протиріччя з вимогами до продуктивності системи. У цьому випадку потрібні зміни проекту, а можливо, і переробка специфікацій.

Для обліку повторюваності етапів процесу розробки створювалися альтернативи каскадної моделі. З таких альтернатив утворилася *ітеративна модель* (рис. 4.2). Ця модель передбачає розбиття життєвого циклу проекту на послідовність ітерацій, кожна з яких нагадує "міні-проект" з усіма фазами життєвого циклу.



Рис. 4.2 – Ітераційна схема розробки програмного забезпечення

Класична ітераційна модель абсолютизує можливість повернень на попередні етапи. Ця обставина відображає істотний аспект програмних розробок: прагнення заздалегідь передбачити всі ситуації використання системи і неможливість в більшості випадків досягти цього. Всі традиційні технології програмування спрямовані лише на те, щоб мінімізувати повернення. Але суть від цього не змінюється: при поверненні завжди доводиться повторювати побудову того, що вже вважалося готовим.

Схема, що підтримує ітераційний характер процесу розробки, була названа схемою з проміжним контролем. Контроль, який виконується за даною схемою після завершення кожного етапу, дозволяє при необхідності повернутися на будь-який рівень і внести необхідні зміни. Основна небезпека використання такої схеми пов'язана з тим, що розробка ніколи не буде завершена, постійно перебуваючи в стані уточнення та вдосконалення.

Мета кожної ітерації в розробці ПЗ – отримання працюючої версії програмної системи, що включає функціональність, визначену інтегрованим змістом усіх попередніх і поточної ітерацій. Результат фінальної ітерації містить всю необхідну функціональність продукту. Таким чином, із завершенням кожної ітерації продукт розвивається інкрементно (нарощує функціональність).

З точки зору структури життєвого циклу і процесу розробки така модель є *ітеративною* ***(iterative)***, а з точки зору розвитку продукту – *інкрементною* ***(incremental)***. Досвід індустрії розробки ПЗ показує, що неможливо розглядати кожен з цих поглядів ізольовано. Тому цю модель часто називають моделлю *ітеративної і інкрементної розробки (Iterative and incremental development, IID)*.

Примітка. Народна мудрість в подібних випадках говорить «краще – ворог хорошого». Залишилося тільки зрозуміти, що можна вважати «хорошим» і як все-таки домогтися кращого ...

Різні варіанти ітераційного підходу реалізовані в більшості сучасних методологій розробки (RUP, MSF, XP).

Стисла характеристика: стадії повторюються неодноразово. Недоліки: система часто погано структурована, проект «непрозорий», після уточнення вимог відкидається частина виконаної раніше роботи; потрібні засоби для швидкого розробки. Використання: підходить для малих і середніх проектів.

 **Спіральна модель.**

Для подолання перелічених проблем в середині 80-х років XX ст. була запропонована спіральна схема (рис. 4.3). Відповідно до даної схеми програмне забезпечення створюється не відразу, а ітераційно з використанням методу прототипування, що базується на створенні прототипів. Саме поява прототипування призвело до того, що процес модифікації програмного забезпечення перестав сприйматися, як «необхідне зло», а став сприйматися як окремий важливий процес.

*Прототипом* називають діючий програмний продукт, який реалізує окремі функції і зовнішні інтерфейси розроблюваного програмного забезпечення.

На першій ітерації, як правило, специфікують, проектують, реалізують і тестують інтерфейс користувача. На другий - додають деякий обмежений набір функцій.

На наступних етапах цей набір розширюють, нарощуючи можливості даного продукту.

Рисунок 4.3 – Спіральна схема розробки програмного забезпечення

Основною перевагою даної схеми є те, що, починаючи з деякої ітерації, на якій забезпечена певна функціональна повнота, продукт можна надавати користувачеві, що дозволяє:

* скоротити час до появи перших версій програмного продукту;
* зацікавити велику кількість користувачів, що забезпечує швидке просування наступних версій продукту на ринку;
* прискорити формування та уточнення специфікацій за рахунок появи практики використання продукту;
* зменшити ймовірність морального старіння системи за час розробки.

Основною проблемою використання спіральної схеми є визначення моментів переходу на наступні стадії. Для її вирішення зазвичай обмежують терміни проходження кожної стадії, ґрунтуючись на експертних оцінках.

У спіральній моделі розроблення програми має вигляд серії послідовних ітерацій. На перших етапах уточнюються специфікації продукту, на наступних – додаються нові можливості функції. *Мета цієї моделі* ***–*** по закінченні кожної ітерації заново здійсніть оцінку ризиків продовження робіт.

На кожному витку спіралі створюють чергову версію продукту, уточнюють вимоги проекту, визначають його якість и планують роботи наступного витка. Особлива увага приділяється початковим етапам розроблення – аналізу і проектуванню, де реалізованість тих чи інших технічних рішень перевіряється і обґрунтовується за допомогою створення прототипів (макетування).

Завдяки ітеративній природі спіральна модель допускає коригування у ході роботи, що сприяє поліпшенню продукту.

При великому числі ітерацій розробка по цій моделі вимагає глибокої автоматизації всіх процесів, інакше вона стає неефективною. На практиці у замовників і користувачів іноді виникає відчуття нестабільності продукту, оскільки вони не встигають стежити за швидкими змінами в ньому.

Спіральна модель розробки ПЗ вимагає визначення ключових точок проекту – *milestones.*

*Виділяють такі основні контрольні точки :*

1. Concept of Operations (COO) – концепція використання системи;
2. Life Cycle Objectives (LCO) – цілі і зміст життєвого циклу;
3. Life Cycle Architecture (LCA) – архітектура життєвого циклу;
4. Initial Operational Capability (IOC) – перша версія ПЗ, перевіряється в ході дослідницької експлуатації;
5. Final Operational Capability (FOC) – готове ПЗ, яке експлуатується в реальних умовах.

*Коротка характеристика спіральної схеми розробки програмного забезпечення:*

− проект має контрольні точки – milestones;

− на кожному витку спіралі створюється прототип;

− на виході – продукт, що відповідає вимогам користувачів.

*Переваги:*

− ранній аналіз можливостей повторного використання;

− наявні механізми досягнення параметрів якості;

− модель дозволяє контролювати джерела проектних робіт і відповідних витрат;

− модель дозволяє вирішувати інтегровані завдання системного розробки, які охоплюють програмну і апаратну складові створюваного продукту.

*Недоліки: п*ісля уточнення вимог відкидається частина раніше виконаної роботи; потрібні кошти для швидкого розробки.

*Використання:* підходить для малих і середніх проектів.

Аналіз в ключових точках особливо актуальний для менеджерів і лідерів проектів, які відстежують хід виконання проекту і планують подальші роботи (рис. 4.4).



Рисунок 4.4 – Розширена спіральна схема розробки програмного забезпечення

Незважаючи на розвиток теоретичної бази, стадія комплексної автоматизації технологій програмування стала можливою лише при відповідному рівні розвитку техніки. Суттєво вплинуло на перехід до комплексної автоматизації усвідомлення того, що не можна розвивати промислове програмування без підтримки технологічних функцій на всіх етапах життя програм. На початку 90 - х рр. з'явилася CASE - технологія (Computer Added Software Engineering – сукупність інструментів і методів програмної інженерії проектування ПЗ для забезпечення високої якості, мінімальної кількості помилок і спрощення проектування), що об'єднувала системи комплексної автоматизації підтримки розробки та супроводу програм.