Міністерство освіти і науки України
Міжнародний гуманітарний університет
Кафедра комп’ютерних наук

**Русу О.П.**

# ВСТУП ДО СПЕЦІАЛЬНОСТІ

Конспект лекцій
для здобувачів вищої освіти,
які навчаються за спеціальностями
галузі «Інформаційні технології»

Одеса 2025

Затверджено Вченою Радою Міжнародного гуманітарного університету.
Протокол № 5 від 20 січня 2025 р.

Русу О.П. Вступ до спеціальності. Конспект лекцій для здобувачів вищої освіти, які навчаються за спеціальностями галузі «Інформаційні технології» [Електронне видання]. Кафедра комп'ютерних наук Міжнародного гуманітарного університету. Одеса, 2025. 39 с.

Дисципліна «Вступ до спеціальності» спрямована на формування у здобувачів цілісного уявлення про сферу професійної діяльності, структуру ІТ-галузі, особливості функціонування ІТ-компаній, основні процеси життєвого циклу програмного забезпечення та ІТ-проєктів, а також правові, етичні та організаційні засади професійної діяльності. У межах дисципліни розглядаються професійні стандарти, нормативні документи, базові підходи до розробки програмних систем, принципи управління проєктами та основи професійного саморозвитку.

Результатом вивчення дисципліни є формування у здобувачів здатності орієнтуватися в структурі сучасної ІТ-галузі, розуміти основні процеси створення та супроводження програмного забезпечення, усвідомлювати роль стандартів і нормативних вимог у професійній діяльності, аналізувати організацію роботи команди розробки та базові принципи управління ІТ-проєктами, а також дотримуватися професійної етики й принципів доброчесності у сфері інформаційних технологій.

**ЗМІСТ**

# ТЕМА 1. ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ЯК ГАЛУЗЬ ПРОФЕСІЙНОЇ ДІЯЛЬНОСТІ

## 1.1 Особливості сучасної ІТ-індустрії

Інженерія програмного забезпечення є однією з ключових галузей сучасних інформаційних технологій, що забезпечує розроблення, впровадження та супроводження програмних систем різного призначення. У сучасному світі програмне забезпечення стало фундаментом функціонування інформаційної інфраструктури держави, бізнесу, науки та освіти. Практично всі сфери діяльності людини пов'язані з використанням комп'ютерних систем, що працюють під керуванням спеціалізованих програмних засобів. Саме тому підготовка фахівців з інженерії програмного забезпечення є важливим напрямом розвитку ІТ-галузі.

Інженерія програмного забезпечення сформувалася як окрема галузь інформатики та програмування у другій половині XX століття. Поява цього напряму була зумовлена стрімким розвитком обчислювальної техніки та збільшенням складності програмних систем. На ранніх етапах розвитку комп'ютерної техніки програмне забезпечення створювалося невеликими групами програмістів і складалося з відносно простих програм. Проте з часом програмні системи почали ускладнюватися, збільшувався обсяг коду, а також зростали вимоги до надійності та ефективності програмних продуктів.

У 1960-х роках виникла проблема, яку у науковій літературі назвали «кризою програмного забезпечення». Вона полягала у тому, що багато програмних проєктів не вдавалося завершити у визначені строки, а створені програмні системи часто містили значну кількість помилок. Саме у цей період сформувалося розуміння необхідності застосування системного підходу до процесу розроблення програмного забезпечення. У 1968 році на конференції НАТО з програмної інженерії було запропоновано термін «software engineering», який підкреслював необхідність застосування інженерних методів до створення програмних систем.

Подальший розвиток інженерії програмного забезпечення пов'язаний із формуванням методологій розроблення програмних систем. На початковому етапі широко застосовувалася каскадна модель життєвого циклу програмного забезпечення, відповідно до якої процес створення програмного продукту поділявся на послідовні етапи: аналіз вимог, проєктування, програмування, тестування та впровадження. Згодом були запропоновані інші моделі життєвого циклу, зокрема ітераційні та інкрементні моделі, а також гнучкі методології розроблення програмного забезпечення, які орієнтовані на швидке реагування на зміну вимог та активну взаємодію з замовником.

У сучасних умовах інженерія програмного забезпечення є комплексною дисципліною, що поєднує знання з програмування, системного аналізу, проєктування програмних систем, тестування, управління проєктами та забезпечення якості програмних продуктів. Фахівці з інженерії програмного забезпечення займаються створенням різноманітних програмних систем, зокрема інформаційних систем підприємств, вебзастосунків, мобільних застосунків, вбудованого програмного забезпечення, систем керування виробничими процесами, а також програмних засобів для оброблення великих обсягів даних.

Програмне забезпечення відіграє надзвичайно важливу роль у сучасній економіці. Більшість підприємств використовує інформаційні системи для автоматизації бізнес-процесів, управління ресурсами та аналізу даних. Банківська система, електронна комерція,

транспортні системи, телекомунікації, охорона здоров'я та багато інших сфер діяльності функціонують завдяки складним програмним комплексам. У промисловості програмне забезпечення використовується для керування виробничими процесами, автоматизації технологічного обладнання, моделювання інженерних систем та оптимізації виробництва.

Важливим аспектом розвитку сучасного суспільства є цифрова трансформація, що передбачає широке використання інформаційних технологій для підвищення ефективності діяльності організацій та покращення якості життя населення. У межах цифрової трансформації впроваджуються електронні державні послуги, розвиваються системи електронного урядування, дистанційної освіти, телемедицини та інші цифрові сервіси. Усі ці системи базуються на використанні програмного забезпечення, що створюється фахівцями з інженерії програмного забезпечення.

ІТ-галузь сьогодні є однією з найбільш динамічних галузей світової економіки. Вона характеризується високими темпами розвитку, значними інвестиціями у дослідження та розроблення, а також постійним зростанням попиту на кваліфікованих фахівців. Багато технологічних компаній займаються створенням програмних продуктів, хмарних сервісів, систем штучного інтелекту та інших інноваційних рішень.

Україна також займає важливе місце на світовому ринку ІТ-послуг. Вітчизняна ІТ-галузь активно розвивається протягом останніх десятиліть і є одним із найбільш динамічних секторів економіки. Українські ІТ-компанії займаються розробленням програмного забезпечення на замовлення, створенням власних програмних продуктів, а також наданням послуг з аутсорсингу та аутстафінгу. Значна кількість українських фахівців працює у міжнародних проєктах, співпрацюючи з провідними технологічними компаніями світу.

Розвиток ІТ-галузі в Україні підтримується високим рівнем технічної освіти та наявністю великої кількості кваліфікованих інженерів-програмістів. Українські університети готують фахівців у галузі програмної інженерії, комп'ютерних наук, інформаційних систем та інших ІТ-спеціальностей. Багато студентів уже під час навчання залучаються до реальних проєктів, що сприяє формуванню практичних навичок розроблення програмного забезпечення.

Сучасна ІТ-галузь характеризується появою нових технологічних напрямів, що суттєво впливають на розвиток програмної інженерії. Одним із таких напрямів є хмарні обчислення, які дають змогу розміщувати програмні системи у розподілених обчислювальних середовищах та забезпечувати доступ до них через мережу Інтернет. Хмарні платформи надають інструменти для зберігання даних, оброблення інформації та розгортання програмних сервісів.

Іншим важливим напрямом є розвиток технологій штучного інтелекту та машинного навчання. Ці технології використовуються для створення інтелектуальних систем, здатних аналізувати великі обсяги даних, розпізнавати зображення та мову, а також приймати рішення на основі отриманої інформації. Штучний інтелект знаходить застосування у багатьох сферах, зокрема у фінансових технологіях, медицині, транспорті та системах безпеки.

Значну роль у розвитку сучасних інформаційних технологій відіграє також Інтернет речей, що передбачає об'єднання великої кількості фізичних пристроїв у єдину мережу для збору та оброблення даних. У таких системах програмне забезпечення забезпечує взаємодію між сенсорами, контролерами, серверними системами та користувацькими застосунками.

Серед інших важливих напрямів розвитку ІТ-галузі можна відзначити кібербезпеку, аналіз великих даних, технології блокчейн, розроблення мобільних застосунків та систем

віртуальної і доповненої реальності. Усі ці напрями потребують висококваліфікованих фахівців з інженерії програмного забезпечення, здатних створювати складні програмні системи та забезпечувати їх надійну роботу.

Таким чином, інженерія програмного забезпечення є важливою складовою сучасної ІТ-галузі, що забезпечує створення програмних систем для різних сфер діяльності. Розвиток інформаційних технологій, цифровізація економіки та поява нових технологічних напрямів зумовлюють постійне зростання ролі програмного забезпечення у житті суспільства. Саме тому підготовка фахівців з інженерії програмного забезпечення є важливим завданням сучасної системи вищої освіти.

### 1.2 Професійні та освітні стандарти у галузі ІТ

Розвиток інженерії програмного забезпечення як професійної галузі діяльності потребує чіткого нормативного та методичного регулювання. Для забезпечення якості програмних продуктів, ефективності процесів розроблення та належного рівня підготовки фахівців використовуються спеціальні професійні й освітні стандарти. Такі стандарти визначають основні вимоги до процесів створення програмного забезпечення, до кваліфікації спеціалістів, а також до структури освітніх програм, за якими здійснюється підготовка майбутніх інженерів програмного забезпечення.

Професійні стандарти у сфері інформаційних технологій встановлюють загальні принципи організації процесів розроблення програмного забезпечення, вимоги до управління програмними проєктами, а також правила забезпечення якості програмних продуктів. Їх використання дає змогу уніфікувати підходи до створення програмних систем, забезпечити сумісність програмних продуктів, підвищити надійність програмного забезпечення та спростити взаємодію між різними організаціями, що беруть участь у розробленні програмних систем.

Важливу роль у формуванні сучасних підходів до інженерії програмного забезпечення відіграють міжнародні стандарти. Одними з найбільш поширених є стандарти, що розробляються міжнародними організаціями ISO (International Organization for Standardization) та IEC (International Electrotechnical Commission). Ці стандарти регламентують різні аспекти життєвого циклу програмного забезпечення, зокрема процеси розроблення, супроводження, тестування та управління програмними системами.

Одним із ключових міжнародних стандартів у галузі програмної інженерії є стандарт ISO/IEC 12207, який визначає структуру процесів життєвого циклу програмного забезпечення. У цьому стандарті описано комплекс процесів, що охоплюють усі етапи створення програмного продукту — від формування вимог до його впровадження та супроводження. Стандарт встановлює базові, допоміжні та організаційні процеси життєвого циклу програмного забезпечення, що дає змогу систематизувати діяльність розробників програмних систем.

Іншим важливим стандартом є ISO/IEC 15504, відомий також як SPICE (Software Process Improvement and Capability Determination). Він використовується для оцінювання зрілості процесів розроблення програмного забезпечення та визначення рівня їх ефективності. Застосування цього стандарту дає змогу організаціям аналізувати власні процеси розроблення програмних систем і визначати напрями їх удосконалення.

У практиці створення програмних систем також широко застосовуються стандарти, пов’язані із забезпеченням якості програмного забезпечення. Наприклад, стандарт ISO/IEC

25010 визначає модель якості програмного продукту, яка містить такі характеристики, як функціональність, надійність, продуктивність, зручність використання, сумісність, безпека та підтримуваність програмної системи. Використання подібних стандартів дає змогу формувати чіткі критерії оцінювання якості програмних продуктів.

Поряд із міжнародними стандартами важливу роль відіграють національні нормативні документи. В Україні нормативна база у сфері інформаційних технологій формується на основі державних стандартів, які гармонізуються з міжнародними стандартами ISO та IEC. Багато міжнародних стандартів у сфері програмної інженерії впроваджуються у вигляді національних стандартів України (ДСТУ), що забезпечує їх використання у практиці вітчизняних організацій.

У сфері освіти стандарти визначають вимоги до підготовки фахівців у галузі інженерії програмного забезпечення. В Україні такі вимоги встановлюються державними стандартами вищої освіти, які визначають компетентності випускників, результати навчання, структуру освітніх програм та загальні вимоги до організації освітнього процесу. Для спеціальності Інженерія програмного забезпечення стандарт вищої освіти визначає перелік професійних знань, умінь та навичок, якими повинні володіти випускники відповідних освітніх програм.

Освітні стандарти також передбачають формування у студентів здатності застосовувати сучасні методи розроблення програмного забезпечення, використовувати інструментальні засоби програмування, здійснювати аналіз вимог до програмних систем, виконувати тестування та забезпечувати якість програмних продуктів. Важливим елементом підготовки майбутніх фахівців є також формування навичок командної роботи, управління програмними проєктами та дотримання професійної етики.

Значний вплив на формування освітніх програм у галузі інженерії програмного забезпечення мають міжнародні рекомендації щодо підготовки IT-фахівців. Одними з найбільш відомих є рекомендації, підготовлені професійними організаціями ACM (Association for Computing Machinery) та IEEE Computer Society. Ці організації розробляють узагальнені моделі навчальних програм для різних напрямів комп’ютерних наук і програмної інженерії.

Зокрема, документ Software Engineering Body of Knowledge (SWEBOK) систематизує основні знання, необхідні для фахівців у галузі програмної інженерії. У цьому документі визначено основні області знань, що охоплюють вимоги до програмного забезпечення, проєктування програмних систем, програмування, тестування, управління конфігурацією, управління проєктами та забезпечення якості програмних продуктів.

Крім того, міжнародні рекомендації щодо структури освітніх програм у галузі програмної інженерії містяться у документах серії Computing Curricula. У цих документах визначаються основні теми навчання, рекомендована структура навчальних курсів, а також співвідношення між теоретичною та практичною підготовкою студентів. Вони використовуються університетами різних країн як орієнтир при розробленні освітніх програм.

Важливим аспектом професійних стандартів у сфері програмної інженерії є також дотримання етичних норм професійної діяльності. Професійні організації розробляють спеціальні кодекси етики, які визначають основні принципи відповідальної діяльності фахівців у галузі інформаційних технологій. Такі кодекси передбачають дотримання принципів чесності, відповідальності, захисту конфіденційності даних та забезпечення безпеки інформаційних систем.

У сучасних умовах розвиток стандартів у сфері інженерії програмного забезпечення є безперервним процесом. Нові технології, поява складних програмних систем, розвиток хмарних обчислень, штучного інтелекту та інших напрямів інформаційних технологій потребують постійного вдосконалення нормативної бази та освітніх програм.

Таким чином, професійні та освітні стандарти відіграють важливу роль у розвитку інженерії програмного забезпечення. Вони забезпечують єдині підходи до організації процесів розроблення програмних систем, визначають вимоги до якості програмного забезпечення та формують основу для підготовки кваліфікованих фахівців у галузі інформаційних технологій. Використання міжнародних і національних стандартів сприяє підвищенню ефективності діяльності ІТ-компаній та забезпечує відповідність підготовки фахівців сучасним вимогам ринку праці.

### 1.3 Організаційна структура ІТ-компаній

ІТ-компанії є складними організаційними системами, діяльність яких спрямована на створення, розвиток та супроводження програмних продуктів і цифрових сервісів. Ефективність функціонування таких організацій значною мірою залежить від правильної побудови їх організаційної структури, розподілу функцій між підрозділами та налагодженої взаємодії між учасниками процесу розроблення програмного забезпечення. Структура ІТ-компанії формується з урахуванням масштабу організації, характеру її діяльності, типу програмних продуктів, що створюються, а також особливостей ринку, на якому працює компанія.

Загалом структура ІТ-компанії містить кілька основних функціональних напрямів діяльності. До них належать управління компанією, розроблення програмного забезпечення, тестування програмних продуктів, проєктування програмних систем, технічна підтримка, маркетинг, продажі та адміністративні служби. Кожен із цих напрямів виконує власні функції, проте всі вони тісно пов’язані між собою у процесі створення програмного продукту.

На найвищому рівні організаційної структури ІТ-компанії знаходиться керівництво організації, яке визначає стратегічні напрями розвитку компанії, формує бізнес-модель, приймає управлінські рішення та контролює виконання основних завдань. До складу керівництва зазвичай входять генеральний директор, технічний директор, директор з розвитку бізнесу та інші керівники ключових напрямів діяльності.

Важливу роль у структурі ІТ-компанії відіграють підрозділи, що безпосередньо займаються створенням програмного забезпечення. Основу цих підрозділів становлять команди розробників програмного забезпечення, до складу яких входять програмісти, тестувальники, системні аналітики, інженери з забезпечення якості, а також фахівці з проєктування програмних систем. У великих компаніях можуть також існувати окремі підрозділи, що відповідають за архітектуру програмних систем, управління конфігурацією програмного забезпечення, а також за підтримку інфраструктури розроблення.

Організаційні моделі ІТ-компаній можуть відрізнятися залежно від способу управління процесом розроблення програмного забезпечення. Однією з поширених моделей є функціональна структура, у межах якої працівники об’єднуються у підрозділи відповідно до їх спеціалізації. Наприклад, окремо функціонують підрозділи програмування, тестування, системного аналізу та технічної підтримки. Така модель дає змогу ефективно

використовувати професійні компетентності спеціалістів, проте може ускладнювати координацію роботи між різними підрозділами.

Іншою поширеною моделлю є проєктна структура організації, за якої формуються окремі команди для реалізації конкретних програмних проєктів. У межах такої команди можуть працювати фахівці різних спеціалізацій — програмісти, тестувальники, аналітики, дизайнери інтерфейсу та менеджери проєктів. Проєктна структура сприяє швидшому прийняттю рішень і більш тісній взаємодії між учасниками розроблення програмного продукту.

У сучасних ІТ-компаніях також широко використовується матрична організаційна структура, яка поєднує елементи функціональної та проєктної моделей управління. У такій структурі працівники одночасно належать до функціональних підрозділів і беруть участь у виконанні конкретних проєктів. Це дає змогу поєднати переваги спеціалізації та гнучкого управління проєктами.

ІТ-компанії можуть відрізнятися не лише внутрішньою структурою, а й типом бізнес-моделі, за якою вони працюють. Одним із поширених типів є аутсорсингові компанії, діяльність яких полягає у виконанні замовлень на розроблення програмного забезпечення для інших організацій. У такому випадку компанія виконує роль виконавця проєкту, а замовник визначає вимоги до програмного продукту. Аутсорсинг дає змогу компаніям зосередитися на власній спеціалізації, зменшити витрати на утримання великих штатів розробників та отримати доступ до міжнародного ринку ІТ-послуг.

Іншим типом ІТ-компаній є продуктові компанії. Вони займаються створенням власних програмних продуктів, які розповсюджуються на ринку у вигляді комерційного програмного забезпечення або цифрових сервісів. Продуктові компанії самостійно визначають функціональні можливості програмних систем, стратегію їх розвитку та способи просування на ринку.

Окрему категорію становлять стартапи — інноваційні компанії, що створюються для реалізації нових технологічних ідей. Стартапи зазвичай мають невелику команду, обмежені ресурси та орієнтуються на швидке створення прототипу програмного продукту і його перевірку на ринку. У разі успішної реалізації ідеї стартап може перетворитися на повноцінну продуктову компанію.

Ще одним типом ІТ-організацій є сервісні компанії, які надають різноманітні ІТ-послуги. До таких послуг можуть належати технічна підтримка програмних систем, системне адміністрування, консалтинг у сфері інформаційних технологій, інтеграція програмних рішень та інші види діяльності.

Важливим елементом діяльності ІТ-компаній є організація роботи команди розробників програмного забезпечення. Команда розроблення програмного продукту зазвичай складається з кількох ролей, кожна з яких виконує певні функції у процесі створення програмної системи. До складу команди можуть входити програмісти, тестувальники, системні аналітики, інженери з якості програмного забезпечення, дизайнери користувацьких інтерфейсів, менеджери проєктів та інші спеціалісти.

Процес розроблення програмного забезпечення передбачає постійну взаємодію між учасниками команди. Системні аналітики формують вимоги до програмної системи на основі потреб замовника. Архітектори програмних систем визначають загальну структуру програмного продукту та принципи його побудови. Програмісти реалізують функціональні можливості програмної системи у вигляді програмного коду. Тестувальники перевіряють правильність роботи програмного забезпечення та виявляють можливі помилки.

У процесі створення програмного продукту важливу роль відіграє також взаємодія між різними підрозділами ІТ-компанії. Наприклад, підрозділ маркетингу аналізує потреби ринку та визначає вимоги до майбутнього програмного продукту. Підрозділ розроблення програмного забезпечення реалізує ці вимоги у вигляді програмної системи. Після завершення розроблення програмний продукт передається до служби технічної підтримки, яка забезпечує його супроводження та допомогу користувачам.

Крім того, у процесі створення програмного забезпечення важливу роль відіграють підрозділи управління проєктами, які відповідають за планування робіт, координацію діяльності команди та контроль виконання завдань. Ефективне управління проєктами дає змогу забезпечити своєчасне завершення робіт, оптимальне використання ресурсів та досягнення поставлених цілей.

Таким чином, структура ІТ-компанії та організація взаємодії між її підрозділами мають важливе значення для ефективного створення програмних продуктів. Використання різних організаційних моделей управління, правильний розподіл функцій між учасниками команди та налагоджена комунікація між підрозділами сприяють підвищенню ефективності процесу розроблення програмного забезпечення та успішній реалізації ІТ-проєктів.

### 1.4 Професійна діяльність інженерів в галузі ІТ

Інженерія програмного забезпечення є однією з ключових галузей сучасних інформаційних технологій, що забезпечує створення програмних систем для різних сфер діяльності суспільства. Фахівці з інженерії програмного забезпечення беруть участь у розробленні програмних продуктів, які використовуються у промисловості, фінансових системах, телекомунікаціях, освіті, медицині, транспорті та багатьох інших галузях. Саме тому професійна діяльність інженера-програміста охоплює широкий спектр завдань, пов’язаних з аналізом вимог до програмних систем, їх проєктуванням, реалізацією, тестуванням та супроводженням.

Основним напрямом діяльності інженера-програміста є розроблення програмного забезпечення. У межах цього напряму фахівець здійснює створення програмних компонентів, реалізацію алгоритмів оброблення даних, розроблення програмних модулів та інтеграцію окремих частин програмної системи. Важливою складовою цієї діяльності є також оптимізація програмного коду, забезпечення ефективності роботи програмних систем та підвищення їх надійності.

Окрім безпосереднього програмування, інженер-програміст бере участь у формуванні вимог до програмного забезпечення. На цьому етапі здійснюється аналіз потреб користувачів або замовника, визначення функціональних можливостей програмної системи та формування технічного завдання. Така діяльність часто виконується спільно з системними аналітиками або архітекторами програмних систем.

Ще одним важливим напрямом професійної діяльності є проєктування програмних систем. У процесі проєктування визначається структура програмного продукту, взаємодія між окремими компонентами програмної системи, а також принципи організації оброблення даних. Проєктування програмних систем передбачає використання різних методів моделювання, структурного та об’єктно-орієнтованого підходів до побудови програмного забезпечення.

Важливою складовою діяльності інженера-програміста є тестування програмного забезпечення. Метою тестування є перевірка правильності роботи програмної системи,

виявлення можливих помилок у програмному коді та забезпечення відповідності програмного продукту встановленим вимогам. Тестування може виконуватися як самими розробниками, так і спеціалізованими фахівцями з контролю якості програмного забезпечення.

Після завершення розроблення програмного продукту важливим етапом є його впровадження та супроводження. У процесі експлуатації програмного забезпечення можуть виникати нові вимоги до системи, необхідність виправлення помилок або вдосконалення функціональних можливостей програмного продукту. Інженери-програмісти беруть участь у модифікації програмних систем, адаптації програмного забезпечення до нових умов використання та підтримці його працездатності.

У сучасних ІТ-компаніях діяльність зі створення програмного забезпечення здійснюється у межах командної роботи. Тому інженер-програміст є частиною команди розроблення програмного продукту, у якій кожен учасник виконує певну функціональну роль. У таких командах можуть працювати фахівці різних спеціалізацій, що забезпечує комплексний підхід до створення програмних систем.

У складі ІТ-команди можна виділити кілька типових функціональних ролей. Однією з основних є роль програміста або розробника програмного забезпечення. Основним завданням такого фахівця є реалізація функціональних можливостей програмного продукту у вигляді програмного коду. Програмісти можуть спеціалізуватися на різних напрямах розроблення програмного забезпечення, зокрема на створенні вебзастосунків, мобільних застосунків, системного програмного забезпечення або вбудованих систем.

Важливу роль у процесі створення програмних систем відіграють системні аналітики. Їх діяльність пов’язана з аналізом вимог до програмного продукту, формуванням технічної документації та визначенням функціональних характеристик програмної системи. Системні аналітики виступають посередниками між замовниками програмного продукту та командою розробників.

Ще однією важливою роллю є архітектор програмного забезпечення. Архітектор визначає загальну структуру програмної системи, принципи взаємодії її компонентів та технологічні рішення, що використовуються у процесі розроблення. Від правильності архітектурних рішень значною мірою залежить ефективність, масштабованість та надійність програмного продукту.

У командах розроблення програмного забезпечення також працюють фахівці з тестування програмних систем. Основним завданням тестувальників є перевірка правильності роботи програмного забезпечення, виявлення дефектів та забезпечення відповідності програмного продукту встановленим вимогам. У сучасній практиці розроблення програмних систем широко застосовуються автоматизовані засоби тестування, що дає змогу підвищити ефективність процесу контролю якості.

Координацію роботи команди розробників здійснюють менеджери проєктів або керівники команд. Вони відповідають за планування робіт, розподіл завдань між учасниками команди, контроль виконання проєкту та взаємодію із замовниками програмного продукту. Ефективне управління проєктом дає змогу забезпечити своєчасне виконання робіт та досягнення поставлених цілей.

Професійна діяльність інженера-програміста базується на певній компетентнісній моделі, що визначає сукупність знань, умінь та професійних якостей, необхідних для виконання професійних завдань. До ключових компетентностей фахівця з інженерії програмного забезпечення належать знання мов програмування, алгоритмів та структур

даних, методів проєктування програмних систем, технологій тестування програмного забезпечення, а також принципів організації командної роботи.

Окрім технічних знань важливе значення мають також так звані універсальні або міжособистісні компетентності. До них належать уміння працювати у команді, здатність до ефективної комунікації, відповідальність за результати роботи, здатність до аналізу складних проблем та пошуку оптимальних рішень. У сучасних умовах важливою компетентністю є також здатність до безперервного професійного розвитку, оскільки технології програмування швидко змінюються.

Кар'єрний розвиток інженера-програміста може здійснюватися за кількома основними напрямами. На початковому етапі фахівець зазвичай займає позицію молодшого програміста, або Junior Software Engineer. На цьому етапі основна увага приділяється набуттю практичного досвіду програмування, засвоєнню інструментальних засобів розроблення та ознайомленню з процесами створення програмних систем.

Після накопичення практичного досвіду фахівець може перейти на позицію програміста середнього рівня, або Middle Software Engineer. На цьому рівні інженер-програміст здатний самостійно виконувати складніші завдання, брати участь у проєктуванні програмних компонентів та відповідати за реалізацію окремих частин програмної системи.

Наступним етапом кар'єрного розвитку є позиція старшого програміста, або Senior Software Engineer. Фахівці цього рівня мають значний досвід роботи, беруть участь у прийнятті технічних рішень, здійснюють наставництво молодших розробників та беруть участь у формуванні архітектури програмних систем.

Подальший розвиток кар'єри може відбуватися у різних напрямах. Частина фахівців обирає технічний напрям розвитку, переходячи на позиції технічного експерта або архітектора програмного забезпечення. Інші фахівці можуть розвиватися у напрямі управління проєктами, займаючи посади керівників команд або менеджерів програмних проєктів.

Вимоги роботодавців до молодих фахівців у сфері інженерії програмного забезпечення постійно зростають. Сучасні IT-компанії очікують від випускників університетів наявності базових знань у галузі програмування, розуміння принципів побудови програмних систем, уміння працювати з системами контролю версій, а також знання основ сучасних технологій розроблення програмного забезпечення.

Важливою вимогою є також здатність до самостійного навчання та швидкого засвоєння нових технологій. Оскільки індустрія програмного забезпечення швидко розвивається, фахівці повинні постійно оновлювати свої знання та опановувати нові інструменти і технології.

Таким чином, професійна діяльність інженера-програміста охоплює широкий спектр завдань, пов'язаних із розробленням, тестуванням та супроводженням програмних систем. Успішна кар'єра у сфері інженерії програмного забезпечення потребує поєднання технічних знань, практичних навичок, здатності працювати у команді та готовності до постійного професійного розвитку. Саме тому підготовка фахівців з інженерії програмного забезпечення спрямована не лише на формування знань у галузі програмування, але й на розвиток професійних компетентностей, необхідних для ефективної діяльності у сучасній IT-індустрії.

# ТЕМА 2. ЖИТТЄВИЙ ЦИКЛ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 2.1 Складові життєвого циклу ПЗ

Розроблення сучасних програмних систем є складним інженерним процесом, який охоплює значну кількість взаємопов'язаних робіт. Для впорядкування діяльності зі створення програмного забезпечення використовується поняття життєвого циклу програмного забезпечення. Життєвий цикл програмного забезпечення описує послідовність процесів і робіт, що виконуються від моменту виникнення ідеї програмного продукту до завершення його використання та зняття з експлуатації. Використання концепції життєвого циклу дає змогу систематизувати діяльність команди розробників, забезпечити контроль якості програмного продукту та ефективно керувати процесом створення програмних систем.

Життєвий цикл програмного забезпечення містить низку етапів, кожен із яких виконує певну функцію у процесі створення програмного продукту. Одним із перших етапів є формування та аналіз вимог. На цьому етапі здійснюється визначення функціональних і нефункціональних характеристик майбутньої програмної системи. Вимоги формуються на основі потреб користувачів або замовника та оформлюються у вигляді технічної документації. Якість формування вимог має вирішальне значення для подальшого успіху проєкту, оскільки саме вимоги визначають функціональні можливості програмного продукту.

Після визначення вимог здійснюється етап проєктування програмної системи. У процесі проєктування формується архітектура програмного продукту, визначаються основні компоненти системи, їх взаємодія та принципи організації оброблення даних. Проєктування програмного забезпечення передбачає створення різних моделей системи, які дають змогу описати її структуру та поведінку. На цьому етапі можуть використовуватися методи структурного та об'єктно-орієнтованого проєктування.

Наступним етапом є реалізація програмного забезпечення, або програмування. У процесі реалізації розробники створюють програмний код, що забезпечує виконання функцій програмної системи. Реалізація програмного продукту зазвичай здійснюється у вигляді окремих програмних модулів, які згодом інтегруються у єдину систему.

Після завершення реалізації програмних компонентів виконується тестування програмного забезпечення. Основною метою тестування є перевірка правильності роботи програмної системи, виявлення помилок у програмному коді та оцінювання відповідності програмного продукту встановленим вимогам. Тестування може включати модульне тестування, інтеграційне тестування, системне тестування та приймальне тестування.

Завершальними етапами життєвого циклу програмного забезпечення є впровадження та супроводження програмного продукту. Під час впровадження програмна система встановлюється у середовищі використання та передається користувачам. У процесі експлуатації програмного забезпечення здійснюється його супроводження, що передбачає виправлення помилок, удосконалення функціональних можливостей системи та адаптацію програмного продукту до змін умов використання.

У межах життєвого циклу програмного забезпечення виконуються різні процеси, які можуть повторюватися багаторазово у вигляді ітерацій. Ітераційний характер розроблення означає, що створення програмного продукту може відбуватися шляхом послідовного

уточнення та вдосконалення програмної системи. У кожній ітерації виконуються роботи з аналізу вимог, проєктування, програмування та тестування, що дає змогу поступово розвивати програмний продукт.

Для організації процесу створення програмного забезпечення використовуються різні моделі життєвого циклу. Модель життєвого циклу визначає порядок виконання етапів розроблення програмної системи та взаємозв'язок між ними. Однією з найвідоміших моделей є каскадна модель життєвого циклу. У межах цієї моделі етапи розроблення виконуються послідовно один за одним. Кожен етап завершується формуванням відповідної документації, після чого розпочинається наступний етап.

Каскадна модель є відносно простою та зрозумілою, проте має певні обмеження. Одним із основних недоліків цієї моделі є низька гнучкість у разі зміни вимог до програмного продукту. Якщо вимоги змінюються після завершення етапу проєктування або реалізації, внесення змін до системи може бути складним і потребувати значних витрат часу.

Для подолання обмежень каскадної моделі були запропоновані інші підходи до організації життєвого циклу програмного забезпечення. Одним із таких підходів є ітераційна модель. У межах ітераційної моделі розроблення програмної системи здійснюється у вигляді послідовності ітерацій, кожна з яких містить повний набір етапів життєвого циклу. Після завершення кожної ітерації формується проміжна версія програмного продукту, яка може бути оцінена користувачами.

Інкрементна модель життєвого циклу передбачає поступове розширення функціональних можливостей програмної системи. У цьому випадку програмний продукт створюється у вигляді послідовності інкрементів, кожен з яких додає нові функції до системи. Такий підхід дає змогу швидше отримати працездатну версію програмного продукту та поступово розвивати його функціональність.

Ще однією відомою моделлю життєвого циклу є спіральна модель. У цій моделі процес розроблення організовано у вигляді послідовності циклів, кожен із яких містить етапи планування, аналізу ризиків, розроблення та оцінювання результатів. Спіральна модель особливо ефективна для складних проєктів, у яких існує значна невизначеність щодо вимог або технологічних рішень.

У сучасній практиці розроблення програмного забезпечення широко застосовуються гнучкі підходи до організації життєвого циклу програмних систем. До таких підходів належать методології, що базуються на принципах гнучкого управління проєктами. Основною ідеєю гнучких підходів є швидка адаптація процесу розроблення до змін вимог, активна взаємодія із замовником та регулярне отримання зворотного зв'язку щодо результатів роботи.

Кожна модель життєвого циклу має власні переваги та обмеження. Каскадна модель є ефективною для проєктів із чітко визначеними вимогами та стабільними умовами реалізації. Ітераційні та інкрементні моделі дають змогу швидше реагувати на зміни вимог і поступово розвивати програмну систему. Спіральна модель дозволяє враховувати ризики у складних проєктах. Гнучкі підходи є особливо ефективними у динамічному середовищі, де вимоги до програмного продукту можуть змінюватися у процесі розроблення.

Процеси життєвого циклу програмного забезпечення регламентуються низкою міжнародних стандартів і рекомендацій. Одним із найбільш відомих стандартів є ISO/IEC 12207, який визначає структуру процесів життєвого циклу програмного забезпечення. Цей стандарт містить опис основних, допоміжних та організаційних процесів, що виконуються під час створення програмного продукту.

У межах життєвого циклу програмного забезпечення важливу роль відіграє аналіз програмного продукту на різних етапах його створення. На етапі формування вимог аналіз спрямований на визначення функціональних характеристик системи. На етапі проєктування здійснюється аналіз архітектурних рішень і структури програмного продукту. Під час тестування проводиться аналіз якості програмного забезпечення та відповідності програмного продукту встановленим вимогам.

Для опису процесів створення програмного забезпечення широко застосовується моделювання життєвого циклу. Моделювання дає змогу формалізувати структуру процесів розроблення програмних систем, визначити взаємозв'язки між різними етапами створення програмного продукту та оптимізувати організацію робіт. У процесі моделювання можуть використовуватися різні методи опису процесів, зокрема діаграми, графічні моделі та спеціалізовані нотації моделювання.

Таким чином, життєвий цикл програмного забезпечення є важливим інструментом організації процесу створення програмних систем. Використання різних моделей життєвого циклу, дотримання міжнародних стандартів та застосування методів моделювання дають змогу підвищити ефективність розроблення програмних продуктів, забезпечити їх якість та адаптувати процес створення програмного забезпечення до вимог сучасної IT-індустрії.

## 2.2 Формування вимог та проєктування програмних систем

Розроблення програмного забезпечення починається з визначення вимог до майбутньої програмної системи. Саме на основі вимог формується уявлення про функціональні можливості програмного продукту, його архітектуру, структуру та принципи роботи. Тому процес збору, аналізу та формалізації вимог є одним із найважливіших етапів життєвого циклу програмного забезпечення. Якість виконання цього етапу значною мірою визначає успішність подальшого розроблення програмної системи.

Вимоги до програмного забезпечення відображають очікування користувачів і замовників щодо функціонування програмного продукту. У загальному випадку вимоги описують, які задачі повинна виконувати система, які обмеження накладаються на її функціонування, а також які характеристики повинні бути забезпечені у процесі експлуатації програмного продукту. Формування вимог передбачає тісну взаємодію між замовниками, користувачами, системними аналітиками та розробниками програмного забезпечення.

Процес збору вимог полягає у виявленні потреб користувачів та формуванні переліку функцій, які повинна виконувати програмна система. Для цього використовуються різні методи, зокрема проведення інтерв'ю з користувачами, аналіз бізнес-процесів, опитування, вивчення існуючих інформаційних систем, а також аналіз документації предметної області. На цьому етапі важливо не лише зібрати інформацію про вимоги, але й забезпечити її коректну інтерпретацію та систематизацію.

Після збору інформації здійснюється аналіз вимог. Основною метою аналізу є уточнення вимог, виявлення можливих суперечностей між ними, визначення пріоритетності окремих функцій та оцінювання можливості реалізації вимог у межах заданих ресурсів. Аналіз вимог дає змогу сформувати узгоджене уявлення про програмну систему та визначити основні характеристики майбутнього програмного продукту.

Важливим результатом процесу аналізу є формування специфікації вимог. Специфікація вимог є документом, у якому систематизовано описано всі вимоги до

програмної системи. Такий документ використовується як основа для подальшого проєктування програмного забезпечення та слугує засобом комунікації між замовниками і командою розробників.

У практиці розроблення програмних систем розрізняють кілька типів вимог. Одним із основних типів є функціональні вимоги. Вони описують функції, які повинна виконувати програмна система, тобто визначають її поведінку у різних ситуаціях. Наприклад, функціональні вимоги можуть описувати оброблення даних, виконання обчислень, формування звітів або взаємодію з користувачем.

Окрім функціональних вимог важливу роль відіграють нефункціональні вимоги. Вони визначають характеристики якості програмної системи, такі як продуктивність, надійність, безпека, зручність використання, сумісність та масштабованість. Нефункціональні вимоги визначають обмеження, у межах яких повинна функціонувати програмна система.

До окремої категорії можна віднести також вимоги до інтерфейсів, які визначають способи взаємодії програмної системи з користувачами, іншими програмними системами або апаратними компонентами. Такі вимоги описують формати обміну даними, протоколи взаємодії та структуру користувацьких інтерфейсів.

Результати аналізу вимог зазвичай оформлюються у вигляді технічного завдання. Технічне завдання є основним документом, що визначає мету створення програмного продукту, його функціональні можливості, вимоги до структури програмної системи, а також умови її розроблення та впровадження. У технічному завданні також можуть визначатися вимоги до апаратного середовища, у якому працюватиме програмна система, вимоги до документації та порядок приймання програмного продукту.

Після формування технічного завдання розпочинається етап проєктування програмної системи. Проєктування передбачає визначення структури програмного продукту, розподіл функцій між його компонентами, а також формування архітектури програмної системи. Основною метою проєктування є створення такої структури програмного забезпечення, яка забезпечує ефективну реалізацію вимог і дає змогу підтримувати програмний продукт у процесі його подальшого розвитку.

У процесі проєктування програмних систем використовуються різні підходи, серед яких найбільш поширеними є структурний та об’єктно-орієнтований підходи. Структурний підхід базується на декомпозиції програмної системи на окремі функціональні компоненти. У межах цього підходу система розглядається як сукупність функцій, кожна з яких виконує певну частину оброблення даних. Для опису структури системи використовуються різні методи функціонального моделювання.

Об’єктно-орієнтований підхід передбачає розгляд програмної системи як сукупності взаємодіючих об’єктів. Кожен об’єкт містить дані та методи їх оброблення. Основними принципами об’єктно-орієнтованого підходу є інкапсуляція, наслідування та поліморфізм. Такий підхід дає змогу створювати гнучкі та масштабовані програмні системи, які легко модифікувати та розширювати.

Одним із ключових завдань етапу проєктування є формування архітектури програмної системи. Архітектура визначає загальну структуру програмного продукту, принципи взаємодії між його компонентами та організацію оброблення даних. Архітектурні рішення визначають технології, що використовуються у процесі розроблення, способи розподілу функцій між компонентами системи та принципи організації програмного коду.

Для опису структури програмних систем широко застосовується моделювання. Моделювання дає змогу створювати формалізовані описи системи, що відображають її

структуру та поведінку. У процесі моделювання можуть використовуватися різні типи моделей, які відображають різні аспекти функціонування програмної системи.

Моделі структури системи описують склад програмних компонентів та зв'язки між ними. Такі моделі дають змогу визначити ієрархію компонентів системи, розподіл функцій між модулями та організацію програмного коду. Моделі поведінки системи відображають порядок виконання операцій у програмній системі, взаємодію між її компонентами та реакцію системи на дії користувачів або зовнішні події.

У процесі проєктування важливе значення має узгодження проєктних рішень із вимогами до програмного забезпечення. Архітектура системи, структура програмних модулів та способи реалізації функцій повинні відповідати вимогам, визначеним на попередніх етапах життєвого циклу програмного забезпечення. Для забезпечення такої відповідності здійснюється перевірка проєктних рішень, аналіз їх ефективності та оцінювання можливості реалізації програмної системи.

Таким чином, процес збору, аналізу та специфікації вимог, а також проєктування програмної системи є ключовими етапами створення програмного забезпечення. Від якості виконання цих етапів залежить ефективність подальшого розроблення програмного продукту, його надійність та відповідність потребам користувачів. Використання системного підходу до формування вимог, застосування сучасних методів проєктування та моделювання програмних систем дає змогу створювати складні програмні продукти, здатні ефективно функціонувати у сучасному інформаційному середовищі.

### 2.3 Конструювання, тестування та забезпечення якості ПЗ

Етап конструювання та реалізації програмного забезпечення є центральною частиною процесу створення програмних систем. Саме на цьому етапі результати аналізу вимог та проєктування перетворюються на конкретні програмні компоненти, що реалізують функціональні можливості системи. Конструювання програмного забезпечення передбачає створення програмного коду, реалізацію алгоритмів, формування структури програмних модулів та забезпечення їх взаємодії у складі єдиної програмної системи.

Конструювання програмного забезпечення здійснюється відповідно до архітектурних рішень, прийнятих на етапі проєктування. Розробники створюють програмні модулі, що реалізують окремі функції системи, дотримуючись визначених принципів організації програмного коду. У сучасній практиці програмування велике значення має модульний підхід, відповідно до якого програмна система розділяється на окремі компоненти, кожен з яких виконує визначену функцію. Такий підхід дає змогу спростити розроблення програмного продукту, підвищити зрозумілість програмного коду та забезпечити можливість повторного використання програмних компонентів.

Під час реалізації програмного забезпечення важливу роль відіграє дотримання правил програмування, використання стандартів оформлення програмного коду та застосування сучасних інструментальних засобів розроблення. Використання систем контролю версій дає змогу організувати ефективну роботу над програмним кодом у команді, відстежувати зміни у програмних модулях та забезпечувати узгодженість різних версій програмного продукту.

Розроблення сучасних програмних систем майже завжди здійснюється у межах командної роботи. Команда розроблення програмного забезпечення може складатися з

програмістів, тестувальників, системних аналітиків, архітекторів програмних систем, фахівців з забезпечення якості програмного забезпечення та менеджерів проєктів. Кожен учасник команди виконує визначені функції, а ефективність процесу розроблення значною мірою залежить від організації взаємодії між членами команди.

Організація командної роботи передбачає планування завдань, розподіл обов'язків між учасниками команди, контроль виконання робіт та координацію діяльності розробників. Для організації спільної роботи використовуються спеціалізовані інструменти керування проєктами, які дають змогу відстежувати виконання завдань, фіксувати зміни у програмному коді та забезпечувати ефективну комунікацію між учасниками команди.

Одним із важливих процесів на етапі реалізації програмного забезпечення є інтеграція програмних компонентів. У процесі інтеграції окремі модулі програмної системи об'єднуються у єдину структуру, що забезпечує виконання всіх функцій програмного продукту. Інтеграція може здійснюватися поступово у міру створення програмних компонентів або після завершення реалізації основних модулів системи.

Під час інтеграції програмних компонентів особлива увага приділяється узгодженню інтерфейсів між модулями та перевірці правильності їх взаємодії. Невідповідність інтерфейсів або помилки у взаємодії компонентів можуть призвести до порушення функціонування програмної системи. Тому процес інтеграції супроводжується ретельним тестуванням програмних модулів та перевіркою їх сумісності.

Важливою складовою процесу створення програмного забезпечення є верифікація та тестування програмних систем. Верифікація передбачає перевірку відповідності програмного продукту встановленим вимогам і технічній документації. Тестування спрямоване на виявлення помилок у програмному коді та оцінювання якості програмного забезпечення.

У практиці розроблення програмних систем застосовуються різні види тестування. Одним із найбільш поширених є модульне тестування. Воно полягає у перевірці окремих програмних модулів на правильність їх функціонування. Модульне тестування зазвичай виконується розробниками програмного забезпечення на етапі реалізації програмного коду.

Після перевірки окремих програмних компонентів виконується інтеграційне тестування. Його метою є перевірка правильності взаємодії між модулями програмної системи. Інтеграційне тестування дає змогу виявити помилки, пов'язані з передаванням даних між компонентами системи або з порушенням логіки їх взаємодії.

Наступним етапом перевірки програмного забезпечення є системне тестування. У процесі системного тестування перевіряється функціонування програмної системи в цілому. На цьому етапі оцінюється відповідність програмного продукту встановленим вимогам, перевіряється робота користувацьких інтерфейсів, а також оцінюється продуктивність та надійність системи.

Окремим видом тестування є приймальне тестування, яке проводиться для підтвердження відповідності програмного продукту вимогам замовника. У процесі приймального тестування користувачі або представники замовника перевіряють функціонування програмної системи та приймають рішення щодо її впровадження.

Важливим завданням у процесі створення програмного забезпечення є забезпечення якості програмного продукту. Якість програмного забезпечення визначається сукупністю характеристик, що відображають здатність програмної системи задовольняти потреби користувачів. До основних характеристик якості програмного продукту належать функціональність, надійність, ефективність, зручність використання, безпека та можливість супроводження.

Для оцінювання якості програмного забезпечення використовуються спеціальні метрики. Метрики якості дають змогу кількісно оцінити різні характеристики програмного продукту, зокрема складність програмного коду, кількість помилок, продуктивність системи, рівень покриття тестами та інші показники. Використання метрик якості дає змогу об'єктивно оцінювати стан програмного продукту та визначати напрями його вдосконалення.

Окрім розроблення та тестування програмного забезпечення важливу роль відіграє документування програмного продукту. Документація є необхідною складовою процесу створення програмних систем, оскільки вона забезпечує збереження інформації про структуру програмного продукту, принципи його роботи та правила використання.

Документування програмного забезпечення здійснюється на різних етапах його створення. На етапі формування вимог створюється документація, що описує функціональні та нефункціональні вимоги до програмної системи. Під час проєктування формується документація, що містить опис архітектури програмного продукту та структури його компонентів.

На етапі реалізації програмного забезпечення важливу роль відіграє документування програмного коду. Коментарі у програмному коді, опис інтерфейсів програмних модулів та технічна документація дають змогу спростити розуміння структури програмної системи та полегшують її подальше супроводження.

Після завершення розроблення програмного продукту створюється користувацька документація, яка містить опис функціональних можливостей програмної системи, правила її встановлення та використання. Така документація призначена для кінцевих користувачів програмного продукту.

Таким чином, конструювання та реалізація програмного забезпечення є складним багатоступеневим процесом, що включає розроблення програмних компонентів, інтеграцію модулів, тестування програмної системи та забезпечення її якості. Використання сучасних методів організації командної роботи, застосування різних видів тестування та належне документування програмного забезпечення дають змогу створювати надійні та ефективні програмні системи, здатні задовольняти потреби користувачів у сучасному інформаційному середовищі.

### 2.4 Впровадження, експлуатація та супроводження ПЗ

Після завершення етапів проєктування, реалізації та тестування програмного забезпечення настає етап впровадження програмного продукту у середовище використання. Впровадження програмного забезпечення передбачає встановлення програмної системи на цільових обчислювальних платформах, налаштування її параметрів, перевірку працездатності у реальних умовах експлуатації та передачу програмного продукту кінцевим користувачам. Цей етап є важливою частиною життєвого циклу програмного забезпечення, оскільки саме під час впровадження відбувається інтеграція програмної системи у інформаційну інфраструктуру організації.

Процес впровадження програмного забезпечення може включати кілька послідовних етапів. На початковому етапі здійснюється підготовка обчислювального середовища, у якому функціонуватиме програмна система. Це може передбачати встановлення необхідного системного програмного забезпечення, налаштування серверів, баз даних, мережевих служб

та інших компонентів інформаційної інфраструктури. Після цього виконується встановлення програмного продукту та його первинне налаштування відповідно до вимог замовника.

Важливою складовою впровадження є перевірка працездатності програмної системи у реальному середовищі використання. На цьому етапі здійснюється тестування системи у робочих умовах, перевіряється правильність взаємодії програмного продукту з іншими інформаційними системами, а також оцінюється продуктивність програмного забезпечення. У разі виявлення недоліків або помилок виконуються необхідні коригування програмної системи.

Після завершення впровадження програмний продукт переходить у стадію експлуатації. Експлуатація програмного забезпечення полягає у його використанні для виконання функцій, передбачених вимогами до програмної системи. У процесі експлуатації програмне забезпечення може обробляти значні обсяги даних, взаємодіяти з іншими інформаційними системами та забезпечувати виконання різноманітних бізнес-процесів.

У ході експлуатації програмного продукту можуть виникати нові вимоги до функціональності системи, необхідність виправлення помилок або адаптації програмного забезпечення до змін умов використання. Для вирішення таких задач здійснюється супроводження програмного продукту. Супроводження програмного забезпечення передбачає виконання комплексу робіт, спрямованих на підтримку працездатності програмної системи, її модернізацію та вдосконалення.

У практиці розроблення програмних систем виділяють кілька основних видів супроводження програмного забезпечення. Одним із них є коригувальне супроводження, яке передбачає виправлення помилок, виявлених під час експлуатації програмного продукту. Такі помилки можуть бути пов’язані з некоректною обробкою даних, порушенням логіки роботи програмної системи або проблемами сумісності з іншими компонентами інформаційної інфраструктури.

Іншим видом супроводження є адаптаційне супроводження. Воно пов’язане з необхідністю адаптації програмного продукту до змін у середовищі його використання. Наприклад, зміни можуть виникати у зв’язку з оновленням операційних систем, апаратного забезпечення або інших програмних компонентів, з якими взаємодіє програмна система.

Також важливим видом супроводження є удосконалювальне супроводження, яке передбачає розширення функціональних можливостей програмного продукту та підвищення його ефективності. У межах такого супроводження можуть додаватися нові функції, оптимізуватися алгоритми оброблення даних або вдосконалюватися користувацький інтерфейс програмної системи.

Важливою складовою процесу супроводження програмного забезпечення є управління змінами програмного продукту. Управління змінами передбачає організацію процесу внесення змін до програмної системи таким чином, щоб забезпечити її стабільну роботу та уникнути появи нових помилок. Для цього застосовуються спеціальні процедури аналізу змін, оцінювання їх впливу на систему та контролю процесу модифікації програмного забезпечення.

Процес управління змінами зазвичай починається з реєстрації запиту на зміну. Такий запит може бути сформований користувачами програмної системи, технічною службою або командою розробників. Після цього здійснюється аналіз запиту, визначається доцільність внесення змін та оцінюється їх вплив на функціонування програмного продукту. У разі прийняття рішення про реалізацію змін вони включаються до плану оновлення програмної системи.

Для забезпечення контрольованого процесу модифікації програмного забезпечення використовується управління конфігурацією. Управління конфігурацією програмного забезпечення передбачає організацію обліку та контролю всіх компонентів програмної системи, зокрема програмного коду, документації, конфігураційних файлів та інших елементів, що входять до складу програмного продукту.

У межах управління конфігурацією здійснюється ідентифікація компонентів програмної системи, контроль змін у цих компонентах, а також відстеження різних версій програмного продукту. Це дає змогу забезпечити узгодженість різних частин програмної системи та уникнути конфліктів під час внесення змін до програмного коду.

Важливим елементом управління конфігурацією є управління версіями програмного забезпечення. У процесі розроблення та супроводження програмного продукту створюються різні версії програмної системи, які можуть відрізнятися набором функціональних можливостей, виправленнями помилок або оптимізацією програмного коду. Управління версіями дає змогу відстежувати історію змін у програмному продукті та забезпечувати можливість повернення до попередніх версій системи у разі необхідності.

Для реалізації управління версіями використовуються спеціалізовані програмні засоби, відомі як системи контролю версій. Такі системи забезпечують збереження різних версій програмного коду, відстеження змін, виконаних розробниками, а також організацію спільної роботи над програмним продуктом. Використання систем контролю версій є необхідною умовою ефективної командної роботи у сучасних ІТ-проєктах.

Системи контролю версій використовують централізовану або розподілену модель зберігання програмного коду. У централізованих системах усі версії програмного коду зберігаються на центральному сервері, до якого звертаються розробники. У розподілених системах кожен учасник команди має локальну копію репозиторію програмного продукту, що дає змогу підвищити гнучкість роботи та забезпечити надійність зберігання даних.

Репозиторій програмного забезпечення є спеціальним сховищем, у якому зберігається програмний код, історія змін, документація та інші компоненти програмної системи. Організація роботи з репозиторіями передбачає використання спеціальних процедур фіксації змін у програмному коді, створення гілок розроблення та об’єднання змін у основну версію програмного продукту.

Використання систем контролю версій дає змогу забезпечити цілісність програмної системи та узгодженість роботи різних розробників. Завдяки таким системам можна відстежувати, які зміни були внесені до програмного коду, хто є автором цих змін та у який момент часу вони були виконані.

Підтримка цілісності та актуальності програмної системи є важливим завданням на етапі експлуатації програмного забезпечення. Цілісність програмної системи означає узгодженість усіх її компонентів та правильність їх взаємодії. Актуальність програмного продукту передбачає своєчасне оновлення програмної системи, виправлення виявлених помилок та адаптацію програмного забезпечення до змін у технологічному середовищі.

Для підтримки актуальності програмної системи використовуються різні механізми оновлення програмного забезпечення, зокрема випуск нових версій програмного продукту, розповсюдження виправлень програмного коду та впровадження нових функціональних можливостей. Регулярне оновлення програмного забезпечення дає змогу підвищити безпеку системи, покращити її продуктивність та забезпечити відповідність сучасним вимогам користувачів.

Таким чином, етапи впровадження, експлуатації та супроводження програмного забезпечення відіграють важливу роль у життєвому циклі програмних систем. Ефективна організація процесів управління змінами, конфігурацією та версіями програмного забезпечення забезпечує стабільну роботу програмного продукту, підтримку його актуальності та можливість подальшого розвитку програмної системи відповідно до потреб користувачів.

# ТЕМА 3. ЖИТТЄВИЙ ЦИКЛ ІТ-ПРОЄКТУ

## 3.1 ІТ-проєкт як форма організації діяльності

У сучасній ІТ-індустрії створення програмних систем, інформаційних сервісів та цифрових продуктів зазвичай організовується у вигляді проєктної діяльності. Саме проєктний підхід дає змогу ефективно планувати роботу команди, раціонально використовувати ресурси та забезпечувати досягнення визначених результатів у встановлені строки. У зв'язку з цим важливим є розуміння сутності ІТ-проєкту, його характеристик, структури управління та ролі учасників проєктної діяльності.

ІТ-проєктом називають комплекс взаємопов'язаних робіт, спрямованих на створення або модернізацію інформаційної системи, програмного продукту чи цифрового сервісу у визначені строки та з використанням обмежених ресурсів. ІТ-проєкт може передбачати розроблення нового програмного забезпечення, впровадження інформаційної системи на підприємстві, створення вебсервісу або модернізацію існуючої програмної системи.

Важливою особливістю ІТ-проєкту є його відмінність від операційної діяльності організації. Операційна діяльність передбачає виконання повторюваних процесів, спрямованих на підтримку функціонування організації. Наприклад, до операційної діяльності можна віднести регулярне обслуговування інформаційних систем, технічну підтримку користувачів або адміністрування серверів. На відміну від цього, проєктна діяльність має тимчасовий характер і спрямована на досягнення конкретного результату. Після завершення проєкту його основні роботи припиняються або переходять у стадію експлуатації створеної системи.

ІТ-проєкти характеризуються кількома важливими властивостями. Однією з основних характеристик є обмеженість у часі. Кожен проєкт має чітко визначений початок і завершення. Терміни виконання робіт зазвичай визначаються у плані проєкту та контролюються у процесі його реалізації.

Ще однією характерною рисою ІТ-проєкту є унікальність результату. Навіть якщо проєкт пов'язаний зі створенням програмного продукту, подібного до вже існуючих систем, він все одно має певні особливості, що відрізняють його від інших проєктів. Унікальність може проявлятися у специфічних вимогах замовника, використанні нових технологій або особливостях середовища використання програмної системи.

Крім того, ІТ-проєкти мають ресурсні обмеження. Для реалізації проєкту використовуються певні ресурси, зокрема фінансові, матеріальні, технічні та людські. Кількість таких ресурсів є обмеженою, тому ефективне управління ними є важливим завданням у процесі виконання проєкту.

Організація робіт у межах ІТ-проєкту здійснюється відповідно до життєвого циклу проєкту. Життєвий цикл ІТ-проєкту описує послідовність етапів, через які проходить проєкт від моменту виникнення ідеї до завершення робіт та передачі результатів замовнику. Розуміння структури життєвого циклу дає змогу організувати процес управління проєктом та забезпечити контроль за виконанням основних завдань.

Першим етапом життєвого циклу ІТ-проєкту є ініціація. На цьому етапі визначається мета проєкту, формулюється загальна ідея майбутнього програмного продукту, аналізується доцільність реалізації проєкту та визначаються основні очікувані результати. У межах етапу

ініціації можуть проводитися попередні дослідження, оцінюватися ризики та визначатися основні учасники проєкту.

Після завершення етапу ініціації розпочинається етап планування. На цьому етапі формується детальний план виконання робіт, визначаються строки реалізації проєкту, розподіляються ресурси та встановлюються контрольні точки виконання завдань. Планування також передбачає визначення структури команди проєкту, розподіл відповідальності між учасниками та формування механізмів контролю виконання робіт.

Наступним етапом є виконання проєкту. Саме на цьому етапі здійснюється реалізація основних робіт, пов'язаних зі створенням програмного продукту. Команда розробників виконує завдання з аналізу вимог, проєктування програмної системи, реалізації програмного коду та тестування програмного забезпечення. У процесі виконання проєкту здійснюється постійна координація роботи команди та взаємодія з замовником.

Паралельно з виконанням робіт здійснюється моніторинг і контроль проєкту. Основною метою цього етапу є відстеження стану виконання робіт, контроль дотримання строків реалізації проєкту та використання ресурсів. У разі виявлення відхилень від плану виконуються коригувальні дії, спрямовані на забезпечення досягнення запланованих результатів.

Завершальним етапом життєвого циклу ІТ-проєкту є завершення проєкту. На цьому етапі здійснюється передача результатів проєкту замовнику, підготовка підсумкової документації та оцінювання результатів виконаної роботи. Також можуть проводитися аналіз виконання проєкту та формування рекомендацій щодо вдосконалення процесів управління проєктами у майбутньому.

У процесі реалізації ІТ-проєкту важливу роль відіграють зацікавлені сторони проєкту. Зацікавленими сторонами або стейкхолдерами називають осіб або організації, які мають інтерес до результатів проєкту або можуть впливати на процес його реалізації. До таких сторін можуть належати замовники проєкту, користувачі програмної системи, керівництво організації, команда розробників, постачальники технологічних рішень та інші учасники проєктної діяльності.

Різні зацікавлені сторони можуть мати різні очікування щодо результатів проєкту. Наприклад, замовники зацікавлені у отриманні програмного продукту, що відповідає їх потребам, а керівництво організації може бути зацікавлене у дотриманні бюджету проєкту та строків його виконання. Тому важливим завданням управління проєктом є врахування інтересів усіх зацікавлених сторін та забезпечення ефективної взаємодії між ними.

Для ефективної реалізації ІТ-проєкту формується організаційна структура управління проєктом. Така структура визначає розподіл ролей та відповідальності між учасниками проєкту, а також механізми взаємодії між ними. У складі організаційної структури зазвичай виділяють керівника проєкту, команду розробників, технічних експертів, представників замовника та інші ролі.

Керівник проєкту відповідає за загальну координацію робіт, планування виконання завдань, управління ресурсами та взаємодію із зацікавленими сторонами. Команда розробників здійснює безпосередню реалізацію програмного продукту, виконуючи роботи з аналізу вимог, програмування, тестування та впровадження програмної системи.

Важливим фактором успішної реалізації ІТ-проєкту є ефективна командна взаємодія. Робота у проєкті передбачає тісну співпрацю між фахівцями різних спеціалізацій, тому важливим є чіткий розподіл ролей та відповідальності між учасниками команди. Кожен член

команди повинен розуміти свої завдання, терміни їх виконання та взаємозв'язок із роботою інших учасників проєкту.

Важливе значення у проєктній діяльності має також організація комунікації. Комунікація передбачає обмін інформацією між учасниками проєкту, узгодження рішень та координацію спільних дій. У процесі реалізації ІТ-проєкту використовуються різні форми комунікації, зокрема робочі наради, звіти про виконання завдань, електронна переписка та використання спеціалізованих систем керування проєктами.

Ефективна комунікація має велике значення не лише всередині проєктної команди, але й у взаємодії зі стейкхолдерами. Регулярне інформування замовників та інших зацікавлених сторін про стан виконання проєкту дає змогу забезпечити прозорість процесу розроблення та своєчасно враховувати зміни вимог до програмного продукту.

Таким чином, ІТ-проєкт є організаційною формою реалізації робіт зі створення програмного забезпечення, що характеризується обмеженістю у часі, унікальністю результату та наявністю ресурсних обмежень. Ефективне управління ІТ-проєктами передбачає чітке планування робіт, визначення ролей учасників проєкту, налагодження взаємодії між членами команди та забезпечення ефективної комунікації зі всіма зацікавленими сторонами. Саме такий підхід дає змогу успішно реалізовувати складні проєкти у сучасній ІТ-індустрії.

### 3.2 Управління ресурсами і ризиками ІТ-проєкту

Планування є одним із ключових процесів управління ІТ-проєктами. Саме на цьому етапі визначаються цілі проєкту, очікувані результати, обсяг робіт, строки виконання, необхідні ресурси та бюджет. Якість планування значною мірою визначає успішність реалізації ІТ-проєкту, оскільки дозволяє заздалегідь оцінити складність робіт, передбачити можливі ризики та організувати ефективну взаємодію учасників команди.

Формування цілей ІТ-проєкту є початковим кроком у процесі планування. Цілі визначають бажаний результат реалізації проєкту та повинні відображати потреби замовника або організації. У контексті розроблення програмного забезпечення цілями можуть бути створення нового програмного продукту, модернізація існуючої інформаційної системи, автоматизація певних бізнес-процесів або розроблення цифрового сервісу. Важливо, щоб цілі проєкту були чітко сформульовані, зрозумілі всім учасникам команди та піддавалися вимірюванню.

Разом із визначенням цілей формуються результати ІТ-проєкту. Результати можуть включати програмний продукт, технічну документацію, створену інформаційну систему або впроваджене програмне рішення. У процесі планування важливо визначити критерії успішності проєкту, які дозволяють оцінити, чи досягнуто поставлених результатів. Такими критеріями можуть бути відповідність функціональних можливостей програмного продукту вимогам замовника, дотримання строків виконання робіт та ефективність використання ресурсів.

Після визначення цілей і результатів проєкту здійснюється визначення обсягу робіт. Обсяг робіт описує всі завдання, які необхідно виконати для досягнення поставлених результатів. У сфері розроблення програмного забезпечення обсяг робіт може включати аналіз вимог, проєктування архітектури програмної системи, реалізацію програмного коду, тестування, впровадження та супроводження програмного продукту. Чітке визначення

обсягу робіт дає змогу уникнути непорозумінь між учасниками проєкту та забезпечити контроль за виконанням завдань.

Для деталізації обсягу робіт часто використовується структуризація робіт проєкту. Така структуризація передбачає поділ загального обсягу робіт на окремі завдання або підпроєкти. Кожне завдання має визначений результат, строки виконання та відповідальних виконавців. Подібний підхід дозволяє спростити планування та контроль виконання проєкту.

Одним із важливих аспектів планування ІТ-проєкту є планування ресурсів. Ресурсами проєкту можуть бути людські ресурси, фінансові кошти, програмне забезпечення, технічне обладнання та інші матеріальні засоби. Планування ресурсів передбачає визначення необхідної кількості спеціалістів, розподіл ролей у команді, оцінювання потреб у технічних засобах та визначення фінансових витрат, необхідних для реалізації проєкту.

Планування строків виконання робіт є ще одним важливим елементом управління проєктом. Для кожного завдання визначаються строки початку та завершення, а також залежності між різними роботами. Наприклад, реалізація програмних модулів може розпочатися лише після завершення етапу проєктування, а тестування програмного продукту виконується після інтеграції програмних компонентів.

Планування бюджету ІТ-проєкту передбачає оцінювання фінансових витрат, необхідних для реалізації проєкту. До бюджету можуть входити витрати на оплату праці розробників, придбання програмного забезпечення, оренду серверного обладнання, використання хмарних сервісів та інші витрати. Формування бюджету дозволяє оцінити економічну доцільність реалізації проєкту та забезпечити контроль за використанням фінансових ресурсів.

Для планування ІТ-проєктів використовуються різні методи та інструменти. Одним із поширених методів є використання мережевих моделей планування, які дозволяють відобразити взаємозв’язки між окремими роботами проєкту. Такі моделі дають змогу визначити послідовність виконання завдань та оцінити загальну тривалість проєкту.

Ще одним поширеним інструментом планування є календарні графіки виконання робіт. У таких графіках відображаються строки виконання завдань, їх взаємозалежність та відповідальні виконавці. Використання календарних графіків дозволяє контролювати хід виконання проєкту та своєчасно виявляти відхилення від плану.

У сучасній практиці управління ІТ-проєктами широко застосовуються спеціалізовані програмні засоби управління проєктами. Такі інструменти дозволяють планувати завдання, контролювати виконання робіт, організовувати комунікацію між членами команди та відстежувати використання ресурсів. До основних функцій таких систем належать управління завданнями, контроль строків виконання робіт, ведення документації проєкту та формування звітності.

На основі результатів планування формується базовий план ІТ-проєкту. Базовий план є основним документом управління проєктом, який містить опис обсягу робіт, календарний графік виконання завдань, план використання ресурсів та бюджет проєкту. Базовий план використовується як основа для контролю виконання проєкту та оцінювання результатів реалізації запланованих робіт.

У процесі реалізації ІТ-проєкту важливе значення має управління ресурсами. Управління ресурсами передбачає раціональний розподіл завдань між членами команди, контроль використання технічних засобів та забезпечення ефективного використання фінансових ресурсів. У разі зміни умов виконання проєкту можуть виникати потреби у коригуванні плану використання ресурсів.

Не менш важливим аспектом управління ІТ-проєктами є управління ризиками. Ризиком у проєктній діяльності називають можливу подію або обставину, яка може негативно вплинути на виконання проєкту. У сфері розроблення програмного забезпечення ризики можуть бути пов'язані з технічними труднощами, зміною вимог замовника, нестачею ресурсів або помилками у плануванні.

Процес управління ризиками включає кілька основних етапів. Першим етапом є ідентифікація ризиків, тобто визначення можливих факторів, які можуть вплинути на реалізацію проєкту. На цьому етапі аналізуються технічні, організаційні та економічні аспекти проєкту.

Після ідентифікації ризиків виконується їх аналіз та оцінювання. Аналіз ризиків дозволяє визначити ймовірність виникнення ризикових ситуацій та можливі наслідки їх реалізації. Оцінювання ризиків дає змогу визначити пріоритетність ризиків та сформувати стратегію їх мінімізації.

Для зменшення впливу ризиків використовуються різні способи мінімізації. До таких способів належать планування резервних ресурсів, використання альтернативних технологічних рішень, поетапне виконання робіт та регулярний контроль стану проєкту. Завдяки таким заходам можна зменшити ймовірність негативного впливу ризиків на результати проєкту.

Важливим елементом навчання управлінню ІТ-проєктами є аналіз прикладів невдалих проєктів. У практиці ІТ-індустрії відомо багато випадків, коли масштабні проєкти завершувалися значними фінансовими втратами або не досягали запланованих результатів. Причинами таких невдач можуть бути недостатнє планування, некоректне визначення вимог, відсутність контролю за виконанням робіт або неефективна комунікація між учасниками проєкту.

Однією з поширених причин провалу ІТ-проєктів є постійна зміна вимог до програмного продукту без належного контролю процесу внесення змін. Іншою причиною може бути недооцінювання складності програмної системи та неправильне планування строків виконання робіт. Також негативний вплив на реалізацію проєктів може мати недостатній рівень взаємодії між учасниками команди або відсутність ефективного управління ризиками.

Аналіз таких прикладів дає змогу визначити типові помилки управління ІТ-проєктами та сформувати рекомендації щодо підвищення ефективності проєктної діяльності. Урахування досвіду попередніх проєктів сприяє вдосконаленню методів планування, підвищенню якості управління ресурсами та покращенню організації роботи команд розробників.

Таким чином, планування ІТ-проєкту є важливою складовою управління процесом розроблення програмного забезпечення. Чітке визначення цілей і результатів проєкту, правильне планування обсягу робіт, ресурсів і строків виконання завдань, використання сучасних інструментів управління проєктами та ефективне управління ризиками забезпечують успішну реалізацію ІТ-проєктів та досягнення запланованих результатів.

### 3.3 Моніторинг, контроль та оцінювання результатів ІТ-проєкту

Ефективне управління ІТ-проєктом передбачає не лише планування та організацію робіт, але й постійний моніторинг і контроль процесу виконання завдань. Моніторинг і

контроль є важливими процесами управління проєктом, які забезпечують своєчасне виявлення відхилень від плану, оцінювання стану виконання робіт та прийняття управлінських рішень, спрямованих на досягнення поставлених цілей. Завдяки цим процесам керівництво проєкту може оцінювати ефективність використання ресурсів, контролювати строки виконання робіт і забезпечувати відповідність результатів проєкту встановленим вимогам.

Моніторинг виконання ІТ-проєкту передбачає систематичне спостереження за перебігом робіт та аналіз отриманих результатів. У процесі моніторингу збирається інформація про стан виконання завдань, використання ресурсів, дотримання строків реалізації проєкту та якість виконаних робіт. Така інформація використовується для оцінювання поточного стану проєкту та визначення необхідності внесення коригувальних дій.

Контроль виконання ІТ-проєкту полягає у порівнянні фактичних результатів виконання робіт із запланованими показниками. У разі виявлення відхилень від плану здійснюється аналіз причин таких відхилень та визначаються заходи щодо їх усунення. Контроль дозволяє забезпечити дотримання основних параметрів проєкту, зокрема строків виконання, бюджету та обсягу робіт.

Для ефективного моніторингу і контролю ІТ-проєктів використовуються різні показники та метрики. Метрики дозволяють кількісно оцінити стан виконання проєкту та об’єктивно визначити ефективність діяльності команди розробників. До таких показників можуть належати строки виконання завдань, рівень використання ресурсів, кількість завершених робіт, а також показники якості програмного забезпечення.

Однією з важливих груп показників є метрики якості програмного забезпечення. Ці метрики використовуються для оцінювання характеристик програмного продукту, зокрема надійності, продуктивності, безпеки та зручності використання. Наприклад, одним із показників якості може бути кількість помилок, виявлених під час тестування програмного продукту або під час його експлуатації. Іншим показником може бути рівень покриття програмного коду тестами, що відображає ступінь перевірки функціональних можливостей системи.

У процесі управління ІТ-проєктами важливу роль відіграють також показники ефективності виконання робіт. Такі показники дозволяють оцінити, наскільки раціонально використовуються ресурси проєкту та чи відповідає фактичний темп виконання робіт запланованим строкам. Аналіз ефективності виконання робіт допомагає керівнику проєкту своєчасно виявляти проблеми та приймати рішення щодо оптимізації процесу реалізації проєкту.

Окрему групу становлять показники продуктивності команди розробників. Продуктивність команди може оцінюватися за різними параметрами, зокрема за кількістю виконаних завдань, швидкістю реалізації функціональних можливостей програмної системи або кількістю помилок у програмному коді. Важливо, щоб оцінювання продуктивності здійснювалося комплексно та враховувало не лише кількісні, але й якісні характеристики виконаної роботи.

У сучасній практиці управління ІТ-проєктами велике значення має оцінювання якості програмного забезпечення безпосередньо у межах реалізації проєкту. Для цього використовуються різні підходи до оцінювання якості, які включають аналіз програмного коду, автоматизоване тестування, перевірку відповідності програмного продукту встановленим вимогам та оцінювання зручності використання програмної системи.

Одним із важливих інструментів забезпечення якості програмного забезпечення є регулярний перегляд результатів виконаних робіт. Такий перегляд може здійснюватися у формі аналізу програмного коду, перевірки виконаних завдань або оцінювання відповідності реалізованих функцій вимогам замовника. Подібні процедури дозволяють своєчасно виявляти недоліки у роботі програмної системи та усувати їх на ранніх етапах розроблення.

У процесі реалізації ІТ-проєкту часто виникає необхідність внесення змін до початкового плану. Причинами змін можуть бути уточнення вимог замовника, виявлення нових технічних обмежень або необхідність адаптації програмної системи до змін у середовищі використання. У зв'язку з цим важливою складовою управління ІТ-проєктом є управління змінами.

Управління змінами передбачає організацію процесу внесення змін до проєкту таким чином, щоб зберегти контроль над реалізацією робіт та уникнути порушення основних параметрів проєкту. Процес управління змінами зазвичай починається з формування запиту на зміну, у якому описується сутність запропонованих змін та їх обґрунтування.

Після отримання запиту виконується аналіз впливу запропонованих змін на строки виконання робіт, бюджет проєкту та обсяг робіт. У разі необхідності зміни можуть бути погоджені із замовником або іншими зацікавленими сторонами. Лише після цього зміни включаються до плану реалізації проєкту.

Важливим аспектом управління ІТ-проєктами є документування управлінських рішень. Документування дозволяє фіксувати ключові рішення, прийняті у процесі реалізації проєкту, а також забезпечує прозорість управління проєктом. Документація може містити опис змін у плані проєкту, протоколи робочих нарад, звіти про стан виконання робіт та інші документи, що відображають перебіг реалізації проєкту.

Наявність належної документації має велике значення для забезпечення ефективної взаємодії між учасниками проєкту та зацікавленими сторонами. Документація дозволяє відстежувати історію прийняття рішень, аналізувати результати виконаних робіт та використовувати накопичений досвід у подальших проєктах.

Особлива роль у процесі управління ІТ-проєктом належить керівнику проєкту. Керівник проєкту відповідає за організацію роботи команди, планування виконання завдань, контроль використання ресурсів та забезпечення досягнення поставлених цілей. Він координує діяльність учасників команди, організовує комунікацію між ними та взаємодіє із замовниками і керівництвом організації.

До основних обов'язків керівника проєкту належать планування робіт, управління ресурсами, контроль строків виконання завдань, аналіз ризиків та забезпечення якості результатів проєкту. Керівник проєкту також відповідає за прийняття управлінських рішень у складних ситуаціях, розв'язання конфліктів у команді та забезпечення ефективної взаємодії між учасниками проєкту.

Відповідальність керівника проєкту включає також забезпечення досягнення запланованих результатів проєкту у межах визначених строків та бюджету. Для цього керівник повинен володіти навичками планування, організації командної роботи, аналізу інформації та прийняття управлінських рішень.

Таким чином, моніторинг і контроль виконання ІТ-проєкту є важливими процесами управління, які забезпечують своєчасне виявлення проблем, оцінювання ефективності роботи команди та підтримку якості програмного продукту. Використання метрик якості та ефективності, організація управління змінами, належне документування управлінських

рішень та відповідальна діяльність керівника проєкту є ключовими умовами успішної реалізації ІТ-проєктів у сучасній ІТ-індустрії.

# ТЕМА 4. ПРОФЕСІЙНА ЕТИКА ТА САМОРОЗВИТОК ІТ-ФАХІВЦЯ

## 4.1 Етичні, соціальні та професійні особливості роботи інженера ПЗ

Розроблення програмного забезпечення є не лише технічною діяльністю, пов'язаною зі створенням програмних систем, але й соціально відповідальним процесом, який відбувається у межах складного економічного, технологічного та соціального середовища. Результати діяльності інженерів програмного забезпечення безпосередньо впливають на функціонування інформаційних систем, роботу підприємств, безпеку даних користувачів та розвиток цифрового суспільства. Саме тому у професійній діяльності фахівців ІТ-галузі важливу роль відіграють етичні норми, принципи професійної доброчесності та відповідальність за результати своєї роботи.

На процес розроблення програмного забезпечення впливає значна кількість чинників, які можна умовно поділити на соціальні, економічні, технологічні та екологічні. Соціальні чинники пов'язані з потребами користувачів, особливостями організації суспільства, рівнем розвитку освіти та культури використання інформаційних технологій. Наприклад, програмні системи повинні враховувати потреби різних груп користувачів, забезпечувати доступність інформаційних сервісів та відповідати принципам інклюзивності.

Економічні чинники визначають можливості фінансування ІТ-проєктів, доступність ресурсів та конкурентні умови на ринку програмного забезпечення. Розроблення програмних систем часто пов'язане з необхідністю оптимізації витрат, підвищення ефективності бізнес-процесів та створення програмних продуктів, які мають комерційну цінність. Економічні умови можуть впливати на вибір технологій, організацію роботи команди розробників та стратегію розвитку програмного продукту.

Технологічні чинники визначають технічні можливості реалізації програмних систем. Вони включають розвиток апаратного забезпечення, появу нових мов програмування, платформ розроблення та інструментальних засобів. Технологічний прогрес створює нові можливості для створення складних інформаційних систем, але водночас вимагає від інженерів постійного оновлення знань і навичок.

Екологічні чинники також можуть впливати на розроблення програмного забезпечення. Хоча програмні системи безпосередньо не пов'язані з фізичним виробництвом, їх використання може впливати на енергоспоживання обчислювальних систем, використання серверних ресурсів та загальний екологічний вплив інформаційної інфраструктури. Тому під час створення програмних продуктів важливим є врахування принципів ефективного використання обчислювальних ресурсів.

Однією з основних моральних засад професійної діяльності інженера програмного забезпечення є доброчесність. Доброчесність передбачає чесність, відповідальність, дотримання професійних стандартів та повагу до прав інших учасників професійної діяльності. У сфері програмної інженерії доброчесність проявляється у відповідальному ставленні до виконання професійних обов'язків, дотриманні авторських прав, уникненні плагіату та коректному використанні програмних компонентів, створених іншими розробниками.

Дотримання принципів доброчесності має особливе значення у сфері розроблення програмного забезпечення, оскільки програмні системи часто використовуються для оброблення конфіденційної інформації, управління важливими технологічними процесами

або надання цифрових послуг широкому колу користувачів. Порушення принципів доброчесності може призвести до серйозних технічних, економічних та соціальних наслідків.

У професійній діяльності інженерів програмного забезпечення можливі також прояви корупції. Корупційні дії можуть проявлятися у неправомірному використанні службового становища, отриманні винагороди за прийняття певних технічних рішень, приховуванні недоліків програмного продукту або маніпулюванні результатами тестування. Подібні дії можуть завдати значної шкоди організації, замовникам програмного продукту та користувачам інформаційних систем.

Корупційні порушення мають не лише правові, але й серйозні репутаційні наслідки. Порушення етичних норм може призвести до втрати довіри з боку роботодавців, партнерів і користувачів програмного забезпечення. У сучасній ІТ-індустрії репутація компанії та її працівників має велике значення, тому дотримання етичних норм є важливою умовою професійної діяльності.

У процесі розроблення програмного забезпечення можуть виникати різні етичні ситуації, що потребують відповідального прийняття рішень. Наприклад, інженер може зіткнутися з ситуацією, коли необхідно обрати між швидким завершенням проєкту та забезпеченням високої якості програмного продукту. Іншою типовою ситуацією є необхідність вирішення питання щодо використання сторонніх програмних компонентів або бібліотек із обмеженими ліцензіями.

Етичні дилеми можуть виникати також у ситуаціях, коли програмне забезпечення використовується для оброблення персональних даних користувачів або коли розробник має доступ до конфіденційної інформації. У таких випадках важливо дотримуватися принципів захисту даних, забезпечення інформаційної безпеки та поваги до прав користувачів.

У межах проєктної діяльності може виникати конфлікт інтересів. Конфлікт інтересів виникає тоді, коли особисті інтереси фахівця можуть впливати на об’єктивність прийняття професійних рішень. Наприклад, конфлікт інтересів може виникнути, якщо розробник одночасно працює над кількома конкуруючими проєктами або має особисту зацікавленість у виборі певних технологічних рішень.

Для запобігання подібним ситуаціям у професійній діяльності ІТ-фахівців використовуються кодекси професійної етики. Кодекс професійної етики інженера програмного забезпечення визначає основні принципи поведінки фахівців у професійній діяльності, їх обов’язки перед суспільством, замовниками та колегами. Такі кодекси формуються професійними організаціями та відображають загальноприйняті норми відповідальної професійної діяльності.

Кодекс професійної етики зазвичай містить принципи відповідальності за якість програмного забезпечення, поваги до прав користувачів, дотримання законодавства та чесності у професійній діяльності. Він також визначає правила взаємодії між членами команди, принципи об’єктивності при прийнятті технічних рішень та необхідність постійного професійного розвитку.

Соціальна відповідальність ІТ-фахівця полягає у врахуванні впливу програмних систем на суспільство. Програмні продукти можуть використовуватися у сфері освіти, медицини, транспорту, фінансових систем та багатьох інших галузях. Помилки у програмному забезпеченні або недбале ставлення до розроблення систем можуть призвести до значних негативних наслідків для користувачів і суспільства в цілому.

Інженер програмного забезпечення несе відповідальність перед замовником програмного продукту за якість виконаної роботи та відповідність програмної системи

встановленим вимогам. Водночас він має враховувати інтереси користувачів програмного забезпечення, забезпечувати безпечність програмних систем та дотримуватися принципів конфіденційності даних.

Професійна відповідальність інженера програмного забезпечення також включає відповідальність за результати розроблення програмного продукту. Це означає, що фахівець повинен виконувати свою роботу на високому професійному рівні, дотримуватися технічних стандартів, використовувати надійні технологічні рішення та забезпечувати належну якість програмного забезпечення.

Крім того, професійна відповідальність передбачає постійне вдосконалення власних знань і навичок. Оскільки інформаційні технології швидко розвиваються, інженер програмного забезпечення повинен постійно оновлювати свої знання, вивчати нові технології та вдосконалювати професійні компетентності.

Таким чином, діяльність інженера програмного забезпечення має не лише технічний, але й важливий соціальний та етичний вимір. Дотримання принципів доброчесності, професійної етики та відповідального ставлення до результатів своєї роботи є необхідною умовою формування довіри до інформаційних технологій та забезпечення їх позитивного впливу на розвиток сучасного суспільства.

### 4.2 Неприпустимість корупції та правові питання використання ПЗ

Розроблення програмного забезпечення є результатом інтелектуальної діяльності людини, тому програмні продукти належать до об’єктів інтелектуальної власності. Захист інтелектуальної власності у сфері програмного забезпечення має важливе значення для забезпечення прав розробників, стимулювання інноваційної діяльності та формування правових засад функціонування ІТ-індустрії. Правове регулювання використання програмного забезпечення визначає правила створення, розповсюдження та використання програмних продуктів, а також встановлює відповідальність за порушення авторських прав.

Одним із основних механізмів захисту програмного забезпечення є авторське право. Програмні продукти у більшості країн світу розглядаються як об’єкти авторського права, аналогічно до літературних або наукових творів. Це означає, що програмний код, алгоритми, структура програмної системи та інші елементи програмного продукту підлягають правовому захисту. Авторське право виникає автоматично з моменту створення програмного продукту і не потребує спеціальної реєстрації, хоча у деяких випадках реєстрація може використовуватися для підтвердження прав автора.

Авторське право на програмне забезпечення надає розробнику або власнику програмного продукту певні юридичні права. Ці права поділяються на майнові та немайнові. Немайнові права пов’язані з особистим зв’язком автора з результатом його інтелектуальної діяльності. До таких прав належать право авторства, право на зазначення імені автора та право на захист твору від спотворення або неправомірного використання.

Майнові права пов’язані з можливістю отримання економічної вигоди від використання програмного продукту. Власник майнових прав має право дозволяти або забороняти використання програмного забезпечення іншими особами, здійснювати його розповсюдження, модифікацію або продаж. Майнові права можуть передаватися іншим особам на підставі договорів або ліцензійних угод.

У сфері розроблення програмного забезпечення важливу роль відіграє ліцензування програмних продуктів. Ліцензія визначає умови використання програмного забезпечення, права та обов'язки користувачів, а також обмеження щодо розповсюдження програмного продукту. Ліцензування є одним із основних механізмів правового регулювання використання програмного забезпечення.

Залежно від умов використання програмного забезпечення розрізняють пропрієтарні та відкриті ліцензії. Пропрієтарні ліцензії передбачають обмеження щодо використання програмного продукту. У цьому випадку програмне забезпечення є власністю розробника або компанії, яка має виключні права на його використання. Користувач отримує право використовувати програмний продукт лише у межах умов, визначених ліцензійною угодою. Наприклад, користувач може мати право встановити програму лише на певну кількість комп'ютерів або використовувати її лише для особистих або комерційних потреб.

На відміну від пропрієтарних ліцензій, відкриті ліцензії передбачають більш широкі можливості використання програмного забезпечення. Програмні продукти, що розповсюджуються за відкритими ліцензіями, можуть бути доступними для перегляду, модифікації та поширення програмного коду. Таке програмне забезпечення зазвичай називають open-source програмним забезпеченням.

Використання open-source програмного забезпечення стало важливою складовою сучасної IT-індустрії. Багато програмних систем, інструментів розроблення та програмних бібліотек створюються та розповсюджуються у межах відкритих проєктів. Участь у таких проєктах дозволяє розробникам обмінюватися досвідом, спільно розвивати програмні рішення та прискорювати розвиток інформаційних технологій.

Водночас використання open-source програмного забезпечення потребує дотримання умов відповідних ліцензій. Відкриті ліцензії можуть встановлювати різні правила використання програмного коду, зокрема вимогу збереження інформації про авторів, обов'язок поширення модифікованих версій програмного продукту на тих самих умовах або обмеження щодо використання програмного забезпечення у комерційних проєктах.

Недотримання умов ліцензій може призвести до порушення авторських прав та юридичної відповідальності. Саме тому інженери програмного забезпечення повинні уважно ставитися до використання сторонніх програмних компонентів і перевіряти умови їх ліцензування.

У сучасній практиці розроблення програмного забезпечення часто використовуються програмні бібліотеки, фреймворки та інші програмні компоненти, створені іншими розробниками. Використання таких компонентів дозволяє значно прискорити процес розроблення програмних систем, проте потребує дотримання правил використання інтелектуальної власності.

Важливою складовою професійної діяльності інженера програмного забезпечення є інформаційна культура. Інформаційна культура передбачає відповідальне ставлення до використання інформаційних ресурсів, дотримання авторських прав, коректне використання джерел інформації та критичний аналіз отриманих даних.

У процесі розроблення програмного забезпечення фахівці часто використовують різні джерела інформації, зокрема технічну документацію, наукові публікації, професійні форуми, навчальні матеріали та приклади програмного коду. Використання таких джерел повинно здійснюватися з дотриманням правил академічної доброчесності та повагою до авторських прав.

Одним із важливих аспектів інформаційної культури є правильне використання професійних джерел інформації. Професійні джерела можуть включати офіційну документацію програмних платформ, стандарти розроблення програмного забезпечення, технічні керівництва та наукові статті. Такі джерела забезпечують достовірну інформацію та допомагають розробникам приймати обґрунтовані технічні рішення.

Важливим правилом роботи з професійними джерелами є перевірка достовірності інформації. У сфері інформаційних технологій велика кількість матеріалів публікується у відкритому доступі, проте не всі джерела є однаково надійними. Тому інженери програмного забезпечення повинні вміти аналізувати джерела інформації, перевіряти їх актуальність та використовувати авторитетні професійні ресурси.

Крім того, важливою складовою інформаційної культури є коректне посилання на використані джерела. Якщо під час розроблення програмного забезпечення використовуються фрагменти програмного коду або технічні рішення, запропоновані іншими розробниками, необхідно дотримуватися умов ліцензії та зазначати авторство відповідних матеріалів.

Таким чином, захист інтелектуальної власності у сфері програмного забезпечення є важливою складовою правового регулювання діяльності IT-індустрії. Авторське право забезпечує захист прав розробників програмних продуктів, а система ліцензування визначає правила використання програмного забезпечення. Дотримання умов ліцензій, відповідальне використання open-source програмного забезпечення та формування високого рівня інформаційної культури є необхідними умовами професійної діяльності інженера програмного забезпечення у сучасному інформаційному суспільстві.

### 4.3 Кар’єрне планування та професійний розвиток в IT-сфері

Сфера інформаційних технологій є однією з найбільш динамічних галузей сучасної економіки. Швидкий розвиток програмних платформ, мов програмування, інструментальних засобів розроблення та інформаційних систем зумовлює постійне зростання попиту на кваліфікованих IT-фахівців. У таких умовах важливим аспектом професійної діяльності є усвідомлене кар’єрне планування, що дозволяє інженеру програмного забезпечення визначати напрями професійного розвитку, формувати необхідні компетентності та поступово досягати нових професійних рівнів.

Кар’єрне планування в IT-галузі передбачає визначення довгострокових професійних цілей, аналіз власних можливостей та формування стратегії професійного зростання. Такий підхід дозволяє фахівцю системно розвивати свої знання та навички, підвищувати кваліфікацію і адаптуватися до змін технологічного середовища. Для молодих спеціалістів кар’єрне планування є особливо важливим, оскільки саме на початкових етапах професійної діяльності формується фундамент майбутнього професійного розвитку.

У сфері інженерії програмного забезпечення існує кілька основних кар’єрних траєкторій. Однією з найбільш поширених є технічна траєкторія розвитку. У межах цього напряму фахівець поступово поглиблює свої технічні знання та спеціалізується на розробленні програмних систем. На початковому етапі кар’єри спеціаліст зазвичай працює на позиції молодшого розробника, де набуває практичного досвіду програмування та знайомиться з процесами створення програмного забезпечення.

Після накопичення досвіду фахівець може перейти на позицію розробника середнього рівня, а згодом — старшого розробника. На цьому етапі інженер бере участь у проєктуванні програмних систем, прийнятті технічних рішень та наставництві молодших спеціалістів. Подальший розвиток у технічному напрямі може передбачати перехід на позицію архітектора програмного забезпечення або технічного експерта.

Іншою кар'єрною траєкторією є управлінський напрям. У цьому випадку фахівець поступово переходить від виконання технічних завдань до управління командою розробників або IT-проєктами. На початковому етапі це може бути роль технічного керівника команди або координатора робіт. Надалі спеціаліст може обіймати посади менеджера проєктів, керівника підрозділу або директора з технологічного розвитку.

Управлінський напрям розвитку передбачає формування компетентностей у сфері управління людьми, планування проєктів, управління ресурсами та організації командної роботи. Фахівці, які обирають цей напрям, повинні володіти не лише технічними знаннями, але й навичками комунікації, лідерства та стратегічного мислення.

Третім напрямом кар'єрного розвитку є експертна спеціалізація. У межах цього напряму інженер програмного забезпечення концентрується на поглибленому вивченні певної технологічної області. Наприклад, фахівець може спеціалізуватися на кібербезпеці, штучному інтелекті, аналізі даних, розробленні вбудованих систем або хмарних технологіях. Експертна спеціалізація дозволяє стати висококваліфікованим фахівцем у певній галузі та брати участь у складних технологічних проєктах.

Важливим фактором формування кар'єри є вимоги ринку праці до молодих фахівців. Сучасні роботодавці очікують від випускників технічних спеціальностей не лише теоретичних знань, але й практичних навичок роботи з сучасними технологіями. До таких навичок належать володіння мовами програмування, розуміння принципів побудови програмних систем, досвід роботи з системами контролю версій, знання основ тестування програмного забезпечення та розуміння процесів розроблення програмних продуктів.

Крім технічних компетентностей важливими є так звані універсальні навички. До них належать уміння працювати у команді, здатність до ефективної комунікації, навички аналізу проблем та прийняття рішень. У сучасних IT-компаніях також високо цінується здатність до самостійного навчання та адаптації до нових технологій.

Важливу роль у професійному розвитку IT-фахівців відіграють професійні спільноти. Професійні спільноти об'єднують спеціалістів, які працюють у певній галузі інформаційних технологій, та створюють умови для обміну досвідом, обговорення технологічних рішень і розвитку професійних контактів. Участь у діяльності таких спільнот сприяє підвищенню професійної компетентності та розширенню кола професійного спілкування.

Ще одним важливим елементом професійного розвитку є участь у конференціях та технічних заходах. IT-конференції, семінари та технічні зустрічі дозволяють ознайомитися з новими технологіями, обговорити сучасні тенденції розвитку інформаційних технологій та встановити контакти з іншими спеціалістами галузі. Подібні заходи часто організовуються професійними організаціями, IT-компаніями або навчальними закладами.

У сфері інформаційних технологій також широко застосовуються професійні сертифікації. Сертифікація підтверджує рівень знань і компетентностей спеціаліста у певній технологічній області. Наявність сертифікатів може підвищувати конкурентоспроможність фахівця на ринку праці та підтверджувати його професійний рівень.

Проте найважливішою умовою професійного розвитку у сфері інформаційних технологій є безперервне навчання. Технології програмування та інформаційні системи

постійно змінюються, тому знання, отримані під час навчання у закладі вищої освіти, повинні постійно оновлюватися. Безперервне навчання передбачає самостійне вивчення нових технологій, проходження професійних курсів, участь у навчальних програмах та використання сучасних освітніх ресурсів.

Формування індивідуальної освітньо-професійної траєкторії розвитку є важливим елементом кар'єрного планування. Така траєкторія визначає послідовність освітніх і професійних кроків, які здійснює фахівець для досягнення своїх професійних цілей. Вона може включати вибір спеціалізації, участь у навчальних програмах, здобуття додаткових сертифікацій, участь у професійних спільнотах та накопичення практичного досвіду.

Індивідуальна освітньо-професійна траєкторія повинна враховувати інтереси фахівця, вимоги ринку праці та перспективи розвитку інформаційних технологій. Такий підхід дозволяє інженеру програмного забезпечення системно формувати власну кар'єру та досягати професійного успіху.

Таким чином, кар'єрне планування у сфері інформаційних технологій є важливим елементом професійної діяльності інженера програмного забезпечення. Визначення кар'єрних траєкторій розвитку, врахування вимог ринку праці, участь у професійних спільнотах, проходження сертифікації та безперервне навчання створюють умови для ефективного професійного зростання та формування висококваліфікованих фахівців у галузі інформаційних технологій.

## РЕКОМЕНДОВАНА ЛІТЕРАТУРА

### Основна


1. Sommerville, Ian. Engineering Software Products: An Introduction to Modern Software Engineering. Pearson, 2020. 342 p. ISBN: 9780135210642.
2. Roger S. Pressman, Bruce R. Maxim. Software Engineering: A Practitioner’s Approach. 2019. 1408 p. ISBN 9781260548006
3. Timinger, Holger. Modern Project Management: Successful Projects with Plan-Based, Agile and Hybrid Approaches. Германія: John Wiley & Sons, Incorporated, 2025. 592p. ISBN: 9783527850631
4. Winters, Titus., Manshreck, Tom., Wright, Hyrum. Software Engineering at Google: Lessons Learned from Programming Over Time. O'Reilly Media, 2020. 602p. ISBN: 9781492082767.
5. І.Л. Бородкіна, Г.О. Бородкін Інженерія програмного забезпечення: Посібник для студентів вищих навчальних закладів. К.: Центр учбової літератури, 2021. 204 с. ISBN 9786110112321.
6. Левус Є. В., Мельник Н. Б. Вступ до інженерії програмного забезпечення: навчальний посібник. Львів: Видавництво Львівської політехніки, 2018. 248 с. ISBN: 9789669411310.


### Допоміжна


7. ISO/IEC/IEEE 12207: Systems and Software Engineering – Software Life Cycle Processes. Швейцарія: IEEE. 2017. ISBN: 9781504442534.
8. Quinn, Michael Jay. Ethics for the Information Age. США: Pearson, 2024. 553. ISBN: 9780138238537.
9. Thomas, David., Hunt, Andrew. The Pragmatic Programmer: Your Journey to Mastery, 20th Anniversary Edition. Великобританія: Pearson Education. 2019. 352p. ISBN 9780135956915.
10. Forsgren, Nicole., Humble, Jez., Kim, Gene. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. США: IT Revolution, 2018. 254p. ISBN: 9781942788331.
11. Fairley, Richard E.. Systems Engineering of Software-Enabled Systems. CIF: Wiley, 2019. 432p. ISBN: 9781119535034.


### Інформаційні ресурси


12. IEEE Computer Society. Software Engineering Body of Knowledge (SWEBOK). URL: https://www.computer.org/education/bodies-of-knowledge/software-engineering
13. PMI – Project Management Institute. URL: https://www.pmi.org/standards/pmbok
14. Scrum Guides – The Scrum Guide. URL: https://scrumguides.org/download.html
15. ISO – ISO/IEC/IEEE 12207:2017. URL: https://www.iso.org/standard/63712.html
16. Git Documentation. URL: https://git-scm.com/docs

Навчальне видання

**Русу Олександр Петрович**

# ВСТУП ДО СПЕЦІАЛЬНОСТІ

Конспект лекцій для здобувачів вищої освіти,
які навчаються за спеціальностями галузі «Інформаційні технології»

Українською мовою