МИКОЛАЇВСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ МЕНЕДЖМЕНТУ
КАФЕДРА ЕКОНОМІЧНОЇ КІБЕРНЕТИКИ,
КОМП'ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Управління ІТ-проєктами

Методичні рекомендації до лекційних занять
для змішаного навчання здобувачів освітнього ступеня «Бакалавр»
за спеціальності F3 (122) «Комп'ютерні науки»
денної та заочної форми здобуття вищої освіти

Миколаїв
2025

УДК 005.8:004
У-66

Друкується за рішенням науково-методичної комісії факультету менеджменту Миколаївського національного аграрного університету (протокол № 2 від 16 жовтня 2025 року)

Укладачі:

С. І. Ємельянов – PhD, старший викладач кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

С. І. Тищенко – в.о. завідувача кафедри, к.п.н., доцент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

О. Ю. Пархоменко – к.ф.-м.н., доцент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

О. О. Жебко – асистент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету;

О. Є. Богатєнкова – асистент кафедри економічної кібернетики, комп'ютерних наук та інформаційних технологій Миколаївського національного аграрного університету

Рецензенти:

Махровська Н. А. – кандидат фізико-математичних наук, доцент кафедри теорії й методики природничо-математичної освіти та інформаційних технологій Миколаївський обласний інститут післядипломної педагогічної освіти

Полянський П. М. – кандидат економічних наук, доцент, доцент кафедри загальнотехнічних дисциплін Миколаївського національного аграрного університету

У-66 **Управління** ІТ-проєктами : методичні рекомендації до лекційних занять для змішаного навчання здобувачів освітнього ступеня «Бакалавр» за спеціальності F3 «Комп'ютерні науки» денної та заочної форми здобуття вищої освіти / уклад. : С. І. Ємельянов, С. І. Тищенко, О. Ю. Пархоменко, О. О. Жебко, О. Є. Богатєнкова. – Миколаїв : МНАУ, 2025. – 80 с.

ЗМІСТ

ПЕРЕДМОВА

У сучасному світі інформаційні технології стали основою функціонування бізнесу, державних структур і повсякденного життя. Створення програмного забезпечення вже давно перестало бути хаотичною творчою діяльністю окремих спеціалістів і перетворилося на складний процес, що потребує системності, точності та узгодженості дій багатьох учасників. Саме тому управління IT-проєктами стало однією з ключових компетенцій сучасного менеджера та фахівця у сфері технологій.

Цей курс лекцій покликаний дати цілісне розуміння процесів, принципів, інструментів і методів, що забезпечують успішне планування, реалізацію та контроль проєктів будь-якої складності. Ми будемо розглядати управління не просто як набір технік або шаблонів документів, а як систему мислення, що об’єднує бізнес-цілі, технологічні рішення та людський фактор у єдиний механізм.

Особливу увагу в курсі приділено практичним аспектам: формуванню проєктної документації, плануванню часових і ресурсних витрат, роботі із зацікавленими сторонами, управлінню ризиками, контролю якості, налагодженню комунікацій у команді та вирішенню конфліктів. Важливо розуміти, що IT-проєкт – це не лише код і технології. Це люди, процеси, обмеження, очікування й рішення, які потрібно приймати щодня.

Курс охоплює як традиційні підходи, так і сучасні методології: від Waterfall до Agile, Scrum і Kanban, а також гібридні моделі, які поєднують стабільність та гнучкість. Це дозволить вам не лише вивчити окремі підходи, а й навчитися обирати оптимальну методологію під реальні умови конкретного проєкту.

Лекції побудовані таким чином, щоб складні теоретичні поняття були зрозумілими й логічними, а кожна тема – не просто академічним матеріалом, а інструментом, який можна застосувати на практиці. Пояснення подаються переважно у форматі розгорнутого тексту з мінімальною кількістю нумерацій, але з акцентами на ключових визначеннях і логічних підтемах. Використання таблиць, прикладів та схем допоможе побачити зв’язки між елементами проєктного менеджменту та краще зрозуміти їх у реальному контексті.

Мета цього курсу – сформувати у студентів здатність не лише знати, а й мислити як менеджер IT-проєктів, приймати виважені рішення, бачити проєкт у цілому, управляти ризиками, комунікаціями та командою. Зрозумівши ці принципи, ви зможете ефективно керувати проєктами будь-якої складності та втілювати технологічні ідеї в реальні продукти, що приносять цінність.

ЗМІСТОВИЙ МОДУЛЬ 1. ОСНОВИ ТА МЕТОДОЛОГІЯ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ

ТЕМА 1.1
ЗАГАЛЬНІ ПОНЯТТЯ І КЛАСИФІКАЦІЯ ПРОЄКТІВ. СТРУКТУРИ УПРАВЛІННЯ ПРОЄКТАМИ

План

1. Поняття проєкту. Ознаки проєкту.
2. Класифікація проєктів за різними критеріями.
3. Життєвий цикл проєкту.
4. Організаційні структури управління проєктами.
5. Роль менеджера проєкту.

Ключові слова: проєкт, класифікація, життєвий цикл, структура, управління, організація, команда, ресурси, менеджер, цілі

Key words: project, classification, life cycle, structure, management, organization, team, resources, manager, goals

1. Поняття проєкту. Ознаки проєкту.

1.1. Поняття проєкту

Проєкт – це тимчасова діяльність, спрямована на створення унікального продукту, послуги або результату. У нього є початок, кінець і чітко визначена мета.

Унікальність є ключовою ознакою. Розробка нового функціоналу для застосунку – унікальна діяльність, навіть якщо команда вже робила щось подібне. Натомість рутинне обслуговування серверів або щоденна технічна підтримка – це операційна діяльність, яка не має завершення й повторюється знову і знову.

1.2. Ознаки проєкту

Основними ознаками проєкту є:

- **Тимчасовість.** Проєкт має визначені дати початку та завершення. Наприклад, компанія планує розгорнути новий модуль у Salesforce протягом трьох місяців.

- **Унікальність результату.** Навіть якщо розробляється чергова версія сайту, саме ця версія – унікальна, адже має специфічні вимоги та особливості.
- **Обмеженість ресурсів.** Команда, бюджет, технології, час – усе це обмежено. У розробці часто трапляється ситуація, коли є 5 розробників, бюджет на 12 спринтів і дедлайн з боку замовника.
- **Наявність життєвого циклу.** Від ініціації до закриття проєкту – кожен етап має свій зміст і результат.
- **Орієнтація на зміни.** Проєкт запускається, коли потрібні зміни – новий продукт, нова можливість, оптимізація витрат чи технологій.
- **Міждисциплінарність.** Часто проєкти об'єднують спеціалістів з різних галузей: розробників, дизайнерів, аналітиків, DevOps-інженерів, тестувальників.

2. Класифікація проєктів за різними критеріями

Проєкти можна класифікувати багатьма способами, і менеджеру важливо розуміти різні підходи, адже вони впливають на вибір структури управління, методології (наприклад, Agile або Waterfall), особливості планування та контролю.

- **За масштабом**
  - Малі – до 6 місяців, невелика команда (4–7 осіб), обмежений бюджет. Приклад: створення лендінгу або чат-бота.
  - Середні – від 6 до 18 місяців, кілька команд, складніша архітектура. Приклад: створення CRM для відділу продажів.
  - Великі – багаторічні проєкти, десятки учасників, складні інтеграції. Приклад: розробка фінансової платформи або ERP-системи для міжнародної корпорації.
- **За напрямом діяльності**
  - IT-проєкти – мобільні застосунки, веб-системи, штучний інтелект, хмарна інфраструктура.
  - Організаційні – зміни бізнес-процесів, впровадження нових структур.
  - Освітні – створення e-learning платформи.
  - Інноваційні – R&D, експерименти з ML-алгоритмами.

У реальному житті проєкти часто поєднують кілька напрямів. Наприклад, впровадження DevOps-практик – одночасно технологічний та організаційний проєкт.

- **За складністю**

    - Прості – мінімальна кількість залежностей, зрозумілі технології. Приклад: створення корпоративного сайту.
    - Складні – багато команд, складна архітектура, інтеграції. Приклад: платформа електронної комерції з мікросервісами.
    - Дуже складні (мегапроєкти) – державні системи, міжнародні інфраструктурні рішення. Приклад: запуск національної системи електронного документообігу.
- **За ступенем визначеності**
    - Добре визначені – вимоги зрозумілі, технології відомі (Waterfall часто підходить). Приклад: перенести локальний сервер у хмару.
    - Слабо визначені (інноваційні) – частину вимог неможливо чітко описати з початку (Agile, Scrum). Приклад: створення AI-функціоналу, який команда досліджує в процесі.
- **За джерелом фінансування**
    - Власні ініціативи компанії – розвиток продукту.
    - Замовні проєкти – аутсорсинг для клієнта.
    - Гранти або державні програми.

Тут важливо розуміти: від джерела фінансування залежить контроль, звітність, а іноді й рівень формальності.

- **За географією команд**
    - Локальні – команда працює в одному офісі.
    - Розподілені – спеціалісти з різних міст чи країн.
    - Глобальні – проєкт веде команда з різних континентів. Приклад: багато українських компаній працюють з клієнтом у США, командою у Польщі та проджектами в Україні.

3. Життєвий цикл проєкту

Життєвий цикл – це шлях, який проходить будь-який проєкт: від появи ідеї до моменту, коли робота завершена, а продукт передано користувачам. У IT цей шлях виглядає як послідовність етапів, де кожен має свої цілі, документи, ризики та результати.

Зазвичай виділяють такі основні фази: ініціація, планування, виконання, моніторинг і контроль, завершення. Але важливо розуміти не назви, а сам зміст.

Ініціація – момент, коли з’являється ідея. Компанія вирішує створити новий мобільний застосунок або оновити CRM. На цьому етапі з'являються базові питання: навіщо це потрібно, яку проблему вирішить продукт, скільки приблизно це коштуватиме, чи є у компанії ресурси.

У реальному житті ініціація часто відбувається швидко. Клієнт пише: «Хочу застосунок як у Uber, тільки для доставки квітів». І менеджер уже на цьому кроці повинен оцінити реалістичність задуму, сформувати візію майбутнього рішення та допомогти замовнику зрозуміти масштаб.

Планування – перетворення ідеї на конкретний план дій. Тут з'являються ключові документи: roadmap, backlog, оцінка трудовитрат, бюджет, графік робіт. Саме на цьому етапі виникають потенційні ризики: наприклад, що інтеграція з банківськими API займе довше, ніж очікувалось, або що тестувальників у команді недостатньо.

Менеджер проєкту на цьому етапі ніби архітектор: він формує «каркас будинку», який команда буде зводити протягом наступних місяців.

Виконання – коли команда створює продукт. Розробники пишуть код, дизайнери створюють макети, тестувальники перевіряють функціонал. В Agile це виглядає як серія ітерацій або спринтів. У Waterfall – як послідовне виконання етапів.

Саме в цій фазі менеджер працює найбільш інтенсивно: організовує зустрічі, прибирає перешкоди, комунікує з клієнтом, контролює дедлайни, координує взаємодію між командою та замовником.

Моніторинг і контроль – оцінка реального прогресу. Завжди існує різниця між тим, що планували на початку, і тим, як ідуть справи зараз. Менеджер контролює якість, бюджет, виконані задачі, ризики й відповідність вимогам.

Уявімо: бекенд запізнюється на два тижні через складність інтеграції. Менеджер не просто «фіксує проблему», а аналізує вплив на загальний графік, пропонує шляхи вирішення – чи перенести частину функціоналу на наступний реліз, чи перерозподілити ресурси.

Завершення – офіційне закриття проєкту. Продукт передано користувачам, команда робить ретроспективу, документує досвід, закриває контракти.

Багато менеджерів недооцінюють важливість цього етапу, але саме він дозволяє компанії вчитися на власних помилках і формувати базу знань.

4. Організаційні структури управління проєктами

Організаційна структура визначає, як саме побудована робота над проєктом, хто приймає рішення та як рухається інформація в команді.

У IT індустрії найчастіше зустрічаються три основні типи структур: функціональна, проєктна та матрична.

Функціональна структура. Усі співробітники належать до своїх відділів: розробники у відділі Engineering, тестувальники – у QA, дизайнери – у Design. Коли виникає новий проєкт, компанія просто «позичає» спеціалістів на деякі задачі.

Це схоже на ситуацію, коли дизайнер одночасно малює банери для маркетингу, оновлює UX у мобільному застосунку та готує презентацію. Він не належить до конкретного проєкту, а працює над кількома напрямами.

Проблема такої структури в тому, що проєкт може просуватися повільно через низький рівень фокусу спеціалістів.

Проєктна структура. Тут все навпаки – формується окрема команда, яка повністю зосереджена на одному проєкті: бекенд, фронтенд, тестувальники, дизайнер, аналітик. Усі вони підпорядковуються менеджеру проєкту. Так працюють багато стартапів або продуктові компанії: є одна команда – і один продукт, над яким вона працює постійно.

Перевага очевидна: фокус, швидкість, злагодженість. Недолік також є: кожен проєкт вимагає окремої команди, а це дорого.

Матрична структура. Найпоширеніша модель у сучасному IT. Спеціалісти одночасно належать і до свого відділу, і до проєкту.

Наприклад, розробник формально працює у відділі Mobile Development, але половина його часу відведена на проєкт з розробки банківського застосунку, а друга половина – на підтримку іншого продукту.

Менеджер проєкту керує роботою на проєкті, а керівник відділу відповідає за розвиток спеціаліста, навчання, зарплату, технічний рівень.

Це гнучка і дещо складна структура, але саме вона дозволяє ефективно використовувати ресурси компанії.

Agile-структури. У компаніях, які працюють за Scrum або іншими гнучкими методологіями, команди зазвичай кросфункціональні. Це означає, що в одній команді є всі ролі, необхідні для доставки продукту: розробники, QA, Product Owner, іноді дизайнер.

Agile-команда працює як самостійна одиниця: вона планує свої спринти, проводить щоденні стендапи, вирішує, як саме реалізувати задачі. Менеджер проєкту тут може виступати у ролі Scrum Master або Delivery Manager.

5. Роль менеджера проєкту

Менеджер проєкту – це не просто людина, яка дивиться у Jira і ставить дедлайни. Це фахівець, який відповідає за результат: за те, щоб продукт був створений, відповідав очікуванням і був доставлений вчасно та в рамках бюджету.

Насправді менеджер – це центральна фігура, що тримає в руках кілька важливих аспектів одночасно:

- **Комунікація.** Менеджер – це «міст» між замовником і командою. Він пояснює клієнту технічні обмеження, а команді – бізнес-логіку задач.

  Уявімо ситуацію: клієнт хоче зробити оплату у два кліки, але команда каже, що це суперечить вимогам безпеки. Менеджер перекладає ці вимоги на мову бізнесу та допомагає знайти компроміс.
- **Управління ризиками.** У будь-якому проєкті щось може піти не так: затримки, зміни вимог, проблеми з інтеграціями. Завдання менеджера – передбачити ризики та підготувати план дій.
- **Управління командою.** Менеджер допомагає підтримувати робочу атмосферу, стежить за навантаженням, розв'язує конфлікти, координує взаємодію між людьми.
- **Контроль бюджету і термінів.** Він не просто слідкує за дедлайнами – він допомагає команді працювати ефективно, а клієнту – не виходити за рамки бюджету.
- **Захист команди.** Гарний менеджер бере на себе спілкування з «важкими» замовниками, фільтрує хаотичні запити та захищає команду від стресу й мікроменеджменту.

У практиці є безліч життєвих моментів:

- ➢ коли клієнт вимагає додати п'ять нових фіч «терміново»,
- ➢ коли дизайнер хворіє, і потрібно терміново знайти заміну,
- ➢ коли тестування затримується, а реліз «горить».

Усі ці ситуації вирішує менеджер. Саме він тримає проєкт у русі.

Питання для самоперевірки

1. Що таке проєкт у контексті управління?
2. Які основні ознаки відрізняють проєкт від операційної діяльності?
3. Що означає термін «унікальність результату» в проєкті?
4. За якими критеріями можуть класифікуватися проєкти?
5. Чим відрізняються технічні проєкти від організаційних?
6. Що таке соціальний проєкт?
7. Які етапи зазвичай включає життєвий цикл проєкту?
8. Чим відрізняється ініціація проєкту від планування?
9. Що відбувається на етапі реалізації проєкту?
10. Навіщо здійснюється моніторинг і контроль у проєкті?
11. Що означає завершення проєкту?

12. Які бувають організаційні структури управління проєктами?
13. Чим функціональна структура відрізняється від матричної?
14. Коли доцільно використовувати проектну (проєктно-орієнтовану) структуру?
15. Які переваги матричної структури управління?
16. Хто такий менеджер проєкту?
17. Які основні обов’язки менеджера проєкту?
18. Чому роль менеджера проєкту є ключовою для успішної реалізації проєкту?

ТЕМА 1.2
МЕТОДОЛОГІЯ УПРАВЛІННЯ ПРОЄКТАМИ

План

1. Поняття методології.
2. Традиційні підходи (Waterfall). Гнучкі методології (Agile, Scrum, Kanban).
3. Сучасні гібридні підходи. Вибір методології для ІТ-проєкту.
4. Практичне застосування методологій.

Ключові слова: методологія, управління, Agile, Scrum, Kanban, Waterfall, гібрид, процес, гнучкість, ефективність

Key words: methodology, management, Agile, Scrum, Kanban, Waterfall, hybrid, process, flexibility, efficiency

1. Поняття методології

1.1. Що таке методологія управління ІТ-проєктами

Поняття методології можна визначити як комплекс принципів, правил, технік, ролей, артефактів та процесів, які формують підхід до виконання проєкту. Методологія задає рамки, у яких команда працює, і водночас створює мову, зрозумілу всім учасникам: менеджерам, розробникам, тестувальникам, аналітикам, дизайнерам, замовникам.

Добре сформована методологія дозволяє уникнути хаосу та непорозумінь. Вона допомагає спланувати роботу, встановити пріоритети, синхронізувати зусилля різних людей та забезпечити контроль над якістю кінцевого продукту. Наприклад, якщо команда використовує гнучку методологію, її робочий процес буде побудований на частих коротких циклах розробки, швидкому реагуванні на зміни та постійній взаємодії з клієнтом. Якщо ж використовується традиційна методологія, робота буде плануватися наперед із чіткою послідовністю етапів.

1.2 Навіщо ІТ-командам потрібна методологія

ІТ-проєкти є складними не лише через технічні аспекти, а й через непередбачуваність, динаміку вимог і кількість зацікавлених сторін. Замовник може не до кінця розуміти, який продукт йому потрібен, ринок може змінюватися в процесі розробки, технології оновлюються щомісяця. За таких

умов методологія виконує роль стабілізатора. Вона дає спосіб планувати та контролювати роботу навіть тоді, коли зовнішні умови нестійкі.

Без методології команди часто стикаються з низкою проблем: невизначеність у пріоритетах, пропуск важливих задач, постійні аврали, дублювання роботи, неправильне трактування вимог. Натомість методологія створює чіткість. Вона визначає хто що робить, у які строки, яким чином контролює виконання, як змінює план, якщо змінюється реальність.

Це можна порівняти з організацією дорожнього руху. Уявімо місто без світлофорів, правил і розмітки: машини рухаються хаотично, аварії неминучі. Але варто впровадити систему правил – і навіть велике місто працює злагоджено. Методологія виконує аналогічну роль у проєктах: вона не гарантує ідеальності, але гарантує **керованість**.

1.3. Які складові формують методологію

Методологія – поняття багатошарове. Вона поєднує в собі філософію роботи (цінності та принципи), інструменти (дошки завдань, артефакти, документи), ролі (менеджер, аналітик, власник продукту, команда розробки), а також процеси (планування, оцінювання, тестування, релізи, комунікація).

Щоб краще зрозуміти структуру методології, погляньмо на таблицю 1.2.1.

Компонент	Суть
Принципи	Основні правила мислення та поведінки, на яких будується підхід (наприклад, прозорість, ітеративність).
Процеси	Послідовність дій, що повторюються під час роботи над продуктом.
Ролі	Хто за що відповідає в команді.
Інструменти	Те, що використовують для організації роботи (дошки, тікети, документація).
Артефакти	Результати процесів: backlog, діаграми, специфікації, прототипи.

Таблиця 1.2.1. Структура методології.

Ця таблиця дає змогу побачити, що методологія – це не щось абстрактне, а цілком реальна структура, яка пронизує всі етапи роботи над проєктом.

1.4. Методологія як елемент командної культури

Ще одна важлива риса: методологія визначає не лише процеси, а й культуру взаємодії. Те, як часто команда спілкується, яким чином відбувається

зворотний зв’язок, як приймаються рішення, наскільки швидко реагують на проблеми – усе це залежить від методології.

Наприклад, у традиційних підходах комунікація здебільшого формальна: звіти, протоколи, узгоджені документи. В Agile культура інша: командна робота, відкритість, короткі статуси, постійне вдосконалення. Методологія формує звички команди, а звички формують результат.

У практиці можна часто побачити ситуацію, коли команда без методології працює реактивно: виникла проблема – усі кидаються її вирішувати. З’явився новий запит – все перевертається догори дриґом. Але коли існує методологія, команда працює проактивно: ризики прогнозуються, зміни плануються, комунікація впорядкована. Це створює відчуття стабільності й знижує стрес.

1.5. Практичне значення методології

Хоча методологія звучить як теоретичне поняття, її практичне значення величезне. Вона допомагає: планувати роботу реалістично, розуміти обсяг задач, оцінювати строки, уникати перевантаження, вчасно виявляти помилки, гармонізувати роботу різних спеціалістів, забезпечувати якість кінцевого продукту.

На простому життєвому прикладі можна уявити ремонт у квартирі. Якщо не мати плану – коли приходять будівельники, хто робить електрику, що потрібно докупити, у якій послідовності виконуються роботи – ремонт розтягнеться на невідомий термін. Але варто сформувати чіткий план і визначити відповідальних – і процес стає контрольованим. Так само й у проєктах: методологія – це план і механізм його виконання.

2. Традиційні підходи (Waterfall). Гнучкі методології (Agile, Scrum, Kanban)

2.1. Традиційний підхід: Waterfall

Традиційна модель управління проєктами, яку часто називають **Waterfall** або «водоспад», – це послідовний підхід до розробки. Він базується на ідеї, що проєкт проходить через чітко встановлені етапи, кожен із яких завершується перед тим, як починається наступний. У водоспаді неможливо перейти до тестування, поки не завершено весь аналіз, не можна почати розробку, поки не затверджено проєктну документацію, і так далі.

Waterfall прийшов у IT зі сфери інженерії та виробництва, де стабільність вимог і точність планування є ключовими. Цей підхід працює добре там, де вимоги чіткі, зрозумілі та малоймовірні до змін, а результат можна детально визначити на старті.

Перевагою Waterfall є те, що водоспад дає ясність. Команда розуміє, що потрібно зробити, які етапи пройти, які ресурси потрібні. Такому проєкту легко скласти бюджет і графік. Замовник також отримує відчуття стабільності – усе розписано та передбачено.

До недоліків Waterfall можна віднести його негнучкість. Якщо замовник змінив думку посеред проєкту або з'явилася нова інформація, повернутися назад дуже складно. Часто буває, що продукт, створений у Waterfall, застаріває ще до релізу. У IT-сфері, де зміни відбуваються швидко, це робить традиційні підходи менш ефективними.

Ситуацію з Waterfall можна порівняти з будівництвом будинку. Спочатку створюється повний план, затверджуються всі рішення, після чого починається будівництво, і вносити зміни вже складно й дорого. Якщо замовник вирішив перенести кухню вже після залиття фундаменту – це майже катастрофа. Тому Waterfall працює там, де зміни небажані або практично неможливі.

2.2. Перехід до гнучкості: чому виник Agile

Традиційні підходи виявилися недостатньо ефективними для IT-проєктів, у яких вимоги постійно змінюються. Це призвело до появи **Agile** – гнучкої філософії управління, яка пропонує зовсім інше мислення. Agile заснований на принципах адаптивності, швидкого отримання зворотного зв'язку та співпраці всередині команди й із замовником.

Agile був сформульований у Маніфесті Agile (2001), де було визначено чотири головні цінності:

- люди та взаємодія важливіші за процеси та інструменти;
- працююче програмне забезпечення важливіше за документацію;
- співпраця із замовником важливіша за контрактні переговори;
- готовність до змін важливіша за слідування плану.

Головна думка Agile полягає в тому, що проєкт має розвиватися поступово, ітераціями, із можливістю корекції на кожному кроці.

2.3. Scrum як найпоширеніша Agile-методологія

Scrum – це не просто процес, а структурована система, яка дозволяє команді працювати швидкими циклами, постійно вдосконалюючись. Scrum організовує розвиток продукту в короткі ітерації, які називаються спринтами. Зазвичай один спринт триває 1–4 тижні.

Scrum визначає ролі (Product Owner, Scrum Master, Development Team), артефакти (Product Backlog, Sprint Backlog, Increment) та ключові події (планування, щоденні стендапи, огляд спринту, ретроспектива). Це створює ритм, у якому команда регулярно показує результат і аналізує свою роботу.

Scrum допомагає швидко отримувати цінність. Якщо замовник хоче щось змінити – це легко врахувати в наступному спринті. Крім того, Scrum створює прозорість: усі бачать прогрес, проблеми й результати.

Це схоже на ремонт квартири, який виконується невеликими зонами: спочатку коридор, потім кухня, потім вітальня. Після завершення кожного етапу власник переглядає результат і може внести корективи, перш ніж команда перейде далі.

2.4. Kanban – підхід без фіксованих циклів

Kanban – гнучкий метод, орієнтований на візуалізацію потоку роботи та зменшення хаосу. На відміну від Scrum, Kanban не має спринтів. Команда працює безперервно, але контролює кількість задач, які виконуються одночасно.

Основним інструментом є Kanban-дошка, поділена на колонки «Заплановано», «В роботі», «Готово» (можуть бути розширені варіанти). Кожна задача – це «картка», яка рухається від стадії до стадії. На дошці видно весь потік роботи, і це дозволяє виявляти затори, перевантаження та проблеми.

Kanban особливо корисний там, де робота надходить нерівномірно або є потреба у швидкому реагуванні: служби підтримки, операційні команди, невеликі проєкти з постійним потоком задач.

Прикладом може бути черга в супермаркеті: коли кас відкрито недостатньо, утворюється затор; коли черги рухаються рівномірно – система працює ефективно. Kanban допомагає керувати такими «чергами» у проєктах.

2.5. Порівняння Waterfall, Scrum і Kanban

Щоб побачити різницю між підходами, розгляньмо їх у таблиці 1.2.2.

Параметр	Waterfall	Scrum	Kanban
Підхід	Послідовний	Ітеративний	Потоковий
Гнучкість	Низька	Висока	Дуже висока
Зміни в роботі	Небажані	Можливі між спринтами	Можливі будь-коли
Планування	Повне на початку	На кожен спринт	Постійне
Доставляння продукту	Наприкінці проєкту	Часто, інкрементами	Постійно
Переваги	Передбачуваність, стабільність	Швидка цінність, прозорість	Гнучкість, мінімальний хаос

Недоліки	Негнучкість, високі ризики запізнілих змін	Потребує дисципліни, командної зрілості	Може бути без структури без контролю

Таблиця 1.2.2. Різницю між підходами до роботи.

3. Сучасні гібридні підходи. Вибір методології для ІТ-проєкту

3.1. Сутність сучасних гібридних підходів

Гібридними називають підходи, які поєднують елементи кількох методологій – найчастіше Waterfall і Agile. Такий підхід дозволяє гнучко реагувати на зміни, але при цьому зберігати чіткість і контроль над ключовими етапами проєкту.

У гібридних моделях можуть співіснувати різні логіки роботи. Наприклад, верхній рівень планування проводиться як у Waterfall, із визначенням архітектури та основних вимог, а реалізація функціоналу виконується ітераційно, за принципами Scrum чи Kanban. У деяких компаніях використовують формальні етапи – аналіз, дизайн, тестування – але всередині кожного етапу команда працює за короткими спринтами.

Сучасні гібридні підходи виникли як відповідь на реальні потреби бізнесу. Редакційні системи, mobile-продукти, банківські платформи, медичне програмне забезпечення та інші складні системи мають як стабільні частини, так і динамічні сегменти, що потребують швидких змін. Одного лише Waterfall або Agile недостатньо, тому команди комбінують елементи, щоб створити оптимальну модель.

3.2. Приклади гібридних підходів

Гібридні моделі можуть бути різними, розглянемо найбільш поширені варіанти.

Waterfall + Scrum. Проєкт планується зверху вниз: мета, архітектура, основні модулі, ключові ризики. Після цього команда працює над розробкою спринтами. Замовник отримує регулярні інкременти продукту, але загальний каркас чітко визначений наперед.

Waterfall + Kanban. Добре підходить для підтримки великих платформ. Стратегічне планування відбувається у формальному стилі, а щоденна робота організовується через Kanban-дошки, з контролем WIP (Work in Progress).

Agile + етапність (Stage-Gate). Особливо популярне в корпораціях. Команда працює ітераційно, але між великими фазами стоять «ворота» – етапи перевірки, після яких приймається рішення «продовжити чи зупинити проєкт».

Dual-Track Agile. Паралельна робота двох потоків: Discovery (дослідження, прототипи, UX) і Delivery (розробка, тестування, реліз). Це дозволяє готувати майбутні фічі, поки поточні вже розробляються.

3.3. Переваги та виклики гібридних моделей

Головна перевага гібридних підходів – можливість отримати «краще з двох світів».

Компанія отримує чітке довгострокове бачення, що приносить Waterfall, але не втрачає адаптивності, яку дає Agile.

Такі моделі роблять процес управління зрозумілішим для бізнесу, оскільки стратегічне планування зберігається, і водночас дають команді свободу швидко реагувати на зміни.

Однак гібридність має і свої виклики. Потрібно правильно визначати, які частини проєкту мають бути гнучкими, а які – стабільними. Також важливо уникати ситуації, коли команда має суперечливі правила: надмірна формальність Waterfall може «зламати» гнучкість Agile-частини, а слабка дисципліна Agile може зробити непередбачуваним планування Waterfall-частини.

3.4. Як вибрати методологію для IT-проєкту

Вибір методології – це не технічне рішення, а стратегічне. На нього впливає природа проєкту, рівень невизначеності, зрілість команди, бюджет, ризики, тип клієнта і навіть корпоративна культура. Тому важливо оцінювати підхід не ізольовано, а в контексті всіх характеристик проєкту.

Нижче подано зведену таблицю 1.2.3., яка допомагає зорієнтуватися.

Характеристика проєкту	Waterfall	Agile	Hybrid
Чіткість вимог	Дуже висока	Низька, змінна	Середня
Ризики змін	Небажані	Постійні	Керовані
Складність продукту	Висока, структурована	Змінна	Комбінована
Очікування клієнта	Статичний контракт	Активна участь	Комбінована взаємодія
Часові рамки	Довгострокові	Гнучкі	Змішані

Придатність	Інженерія, державні проєкти	Стартапи, інновації	Великі IT-платформи

Таблиця 1.2.3. Вибір метолодогії від потреб проєкту.

Щоб обрати оптимальний підхід, менеджер може рухатися за кількома кроками.

- **Оцінити невизначеність**. Якщо вимоги нестабільні, Agile чи гібрид є кращими. Якщо стабільні – можна планувати як Waterfall.
- **Оцінити команду**. Зріла команда, яка вміє самоорганізовуватися, краще працює в Agile. Команди-початківці можуть потребувати структурованості Waterfall.
- **Врахувати клієнта**. Якщо клієнт хоче бачити результат поступово – Agile. Якщо хоче чіткий контракт – Waterfall або Hybrid.
- **Оцінити масштаби і критичність продукту**. У складних платформах часто потрібні гібриди.
- **Проаналізувати ризики**. Agile краще для інновацій, Waterfall – для передбачуваних систем.

Ця логіка нагадує вибір інструменту в майстерні. Не можна всі завдання виконати молотком або лише викруткою – потрібно обирати інструмент під конкретну задачу.

4. Практичне застосування методологій.

4.1. Впровадження методології: від теорії до практики

Перший крок – це розуміння, що методологія не починає працювати автоматично. Її впровадження – це процес, який потребує підготовки команди, комунікації, тренувань та адаптації. Нерідко компанії стикаються з помилкою, коли «встановлюють Scrum» або «переходять на Kanban», але не змінюють мислення та культуру. Практика показує, що методологія працює лише тоді, коли команда розуміє, навіщо вона їй потрібна, і коли кожен учасник приймає новий спосіб роботи.

Уявімо ситуацію: компанія вирішила «стати Agile», але при цьому керівництво продовжує вимагати довгострокові плани на 12 місяців і докладні звіти тричі на день. У такій ситуації Agile не запрацює, оскільки практики не співпадають зі змінами у мисленні. Практичне застосування методології – це передусім зміна поведінки.

4.2. Використання Waterfall на практиці

Традиційні методології, зокрема Waterfall, застосовуються у проєктах із чіткими вимогами, стабільним середовищем і високою регуляцією. У таких

умовах потрібно заздалегідь знати, що саме буде створено, у які строки й у якій послідовності.

Прикладом можуть бути державні проєкти або проєкти у фінансовій сфері, де кожна зміна вимог проходить складну бюрократичну процедуру. У таких випадках покроковий підхід є виправданим. На практиці робота будується так: спочатку проводиться детальний аналіз і створюється велика документація, потім формується архітектура, після – розробка, тестування та передача замовнику. Усі етапи логічно послідовні.

Практична цінність Waterfall полягає в можливості ретельного контролю. Кожен етап має чіткий результат, який можна зафіксувати й перевірити. Проте на практиці Waterfall не підходить для динамічних проєктів – тому у більшості IT-компаній він використовується точково, лише там, де стабільність важливіша за швидкість.

4.3. Застосування Scrum у реальних командах

Scrum – найпоширеніша на практиці методологія у сфері розробки програмного забезпечення. У реальному середовищі Scrum створює ритм, у якому команда працює короткими ітераціями, регулярно демонструє результати й аналізує свою роботу.

На практиці це виглядає так: команда планує спринт, обирає задачі із беклогу, щоденно синхронізується на стендапі, наприкінці спринту демонструє створений інкремент продукту, а потім проводить ретроспективу. Усе це повторюється, доки продукт не буде готовий.

Основна сила Scrum у практичному застосуванні – це прозорість процесів та регулярний зворотний зв’язок. Команда бачить свій прогрес, замовник – отримує частинами те, що хоче, а менеджер – контролює ризики.

4.4. Практика використання Kanban

Kanban часто застосовується у командах підтримки, DevOps-командах, у технічних відділах, де задачі приходять нерівномірно. На практиці це найбільш «легка» й найменш формалізована методологія. У Kanban немає спринтів, немає ролей, немає суворих правил. Головний інструмент – це візуалізація потоку роботи.

Команда створює Kanban-дошку, де кожна колонка символізує певний етап роботи. Задачі рухаються по дошці, і команда бачить, де виникають затримки. Наприклад, якщо у колонці «Testing» накопичилося багато задач – це сигнал, що тестувальники перевантажені або процес непрацюючий.

Практична цінність Kanban – це гнучкість, мінімальний хаос і швидка реакція на зміну навантаження.

Характеристика	Waterfall	Scrum	Kanban
Тип проєктів	Стабільні, регульовані	Змінні, продуктові	Нерівномірні, операційні
Стиль роботи	Послідовний	Ітеративний	Потоковий
Зміни вимог	Небажані	Часті	Будь-коли
Інструменти	Документація, діаграми	Backlog, спринти, стендапи	Kanban-дошка
Ризики	Зміни дорогі	Потребує дисципліни	Потребує самоконтролю
Цінність	Передбачуваність	Прозорість + швидка доставка	Гнучкість і контроль навантаження

Таблиця 1.2.4. Порівняння практичного застосування Waterfall, Scrum і Kanban.

У сучасних IT-проєктах дуже часто використовують гібридні моделі. Наприклад, розробка продукту може вестися за Scrum, а технічна підтримка – за Kanban. Деякі компанії формують гібридні моделі на рівні команд, де частина процесів чітко спланована (як у Waterfall), а частина – ітеративна (як у Agile).

Такі підходи дозволяють отримати переваги різних методологій. Наприклад, у великому банківському проєкті вимоги та архітектура можуть плануватися за Waterfall, а щоденна розробка інтерфейсів – за Scrum. Це робить процес керованим, але водночас адаптивним.

4.5. Типові проблеми та рішення при практичному застосуванні методологій

Найпоширеніші труднощі виникають тоді, коли методологія застосовується формально. Команда проводить стендапи, але не обмінюється корисною інформацією. Дошку Kanban створюють, але не оновлюють. Планування спринтів роблять, але беклог не пріоритезують. У таких випадках методологія втрачає сенс.

Практика показує, що головним фактором успіху є зрілість команди та готовність до змін. Коли команда приймає методологію, а не «імітує» її, процеси працюють і дають результат. Менеджер у цьому випадку стає фасилітатором – людиною, яка допомагає команді працювати краще.

Питання для самоперевірки

1. Що таке методологія управління проєктами?
2. Навіщо проєктам потрібна методологія?
3. Які основні ознаки традиційних підходів до управління проєктами?
4. Що таке модель Waterfall?
5. У чому полягає головний недолік Waterfall?
6. Що означає термін Agile?
7. Які принципи лежать в основі гнучких методологій?
8. Чим Scrum відрізняється від інших Agile-підходів?
9. Які ролі входять до Scrum-команди?
10. Що таке Kanban і яку проблему він допомагає вирішити?
11. Чим Kanban-дошка корисна для команди?
12. Що означає поняття «гібридна методологія»?
13. У яких випадках доцільно поєднувати Waterfall та Agile?
14. Які фактори впливають на вибір методології для IT-проєкту?
15. Чому важливо враховувати вимоги замовника при виборі методології?
16. Які ризики можуть виникнути при неправильному виборі методології?
17. Як методології допомагають організувати роботу команди?
18. Чому практичне застосування методологій важливіше, ніж теоретичне знання?

ЗМІСТОВИЙ МОДУЛЬ 2
ІНІЦІАЦІЯ ТА ПЛАНУВАННЯ ІТ-ПРОЄКТІВ

ТЕМА 2.1
ІНІЦІАЦІЯ ПРОЄКТУ. ПАСПОРТ ПРОЄКТУ

План

1. Стадія ініціації.
2. Формулювання цілей і завдань.
3. Зацікавлені сторони.
4. Паспорт проєкту: складові та значення.
5. Критерії успіху. Обґрунтування доцільності.

Ключові слова: ініціація, паспорт, цілі, завдання, зацікавлені сторони, критерії, успіх, обґрунтування, старт, стратегія

Key words: initiation, charter, goals, tasks, stakeholders, criteria, success, justification, start, strategy

1. Стадія ініціації

1.1. Стадія ініціації: зміст і головна мета

Стадія ініціації – це початковий етап життєвого циклу проєкту, на якому відбувається формальне та концептуальне обґрунтування його існування. На цьому етапі потрібно відповісти на ключові питання: Для чого існує проєкт? Яку цінність він створює? Хто його замовник? Який бажаний результат? Хто буде брати участь у роботі?

Головна мета ініціації – офіційно запустити проєкт, надавши менеджеру чіткі рамки, обґрунтування та мандат на подальшу роботу. Ініціація створює основу: якщо вона проведена правильно, усі наступні кроки – планування, виконання, контроль – стають значно впорядкованішими.

Будь-який проєкт має починатися з конкретної бізнес-проблеми або можливості. Це може бути необхідність покращити внутрішню систему, створити новий цифровий продукт або оптимізувати бізнес-процес. Якщо немає чіткого розуміння проблеми, команда ризикує витрачати ресурси на створення того, що нікому не потрібно.

Простий життєвий приклад: якщо будинок має протікаючий дах, то очевидно, що його потрібно ремонтувати. Але якщо причина протікання не визначена (наприклад, винний не дах, а труба кондиціонера), то будь-який

ремонт буде неефективним. Так само й у проєктах: чітке визначення бізнес-потреби дозволяє сформувати правильне бачення рішення.

1.2. Основні елементи стадії ініціації

Ініціація включає кілька ключових компонентів, кожен з яких формує основу для майбутнього планування.

Визначення зацікавлених сторін (stakeholders). Це всі, хто впливатиме на проєкт або відчуватиме його наслідки: замовник, менеджери, технічні лідери, кінцеві користувачі, інвестори. Важливо не лише ідентифікувати їх, а й зрозуміти їхні очікування, інтереси та рівень впливу.

Формування початкового бачення проєкту. На цьому етапі створюється загальний опис того, що потрібно зробити. Це не деталізована специфікація, а короткий концепт – Project Vision. Він допомагає всім служити одній ідеї.

Оцінка можливостей і ризиків. Попередня оцінка ресурсів, бюджетів, часових обмежень і потенційних перешкод. Це дає змогу зрозуміти, чи варто взагалі запускати проєкт.

Підготовка документа ініціації (Project Charter). Це формальний документ, який затверджує запуск проєкту. У ньому визначається: мета, опис, зацікавлені сторони, обмеження, очікувані результати, відповідальні особи.

Для структурування цього етапу можна використати таблицю 2.1.1.

Елемент ініціації	Що містить
Бізнес-потреба	Проблема або можливість
Зацікавлені сторони	Перелік і опис впливу
Початкове бачення	Що створюється і для кого
Ризики	Потенційні перешкоди
Обмеження	Бюджет, час, технології
Project Charter	Формальний старт проєкту

Таблиця 2.1.1. Документ ініціації (Project Charter).

1.3. Вибудова логіки проєкту: від ініціації до формулювання цілей

Після того як проєкт отримує офіційний старт, наступний етап – чітке визначення того, що саме потрібно досягти. Формулювання цілей – один із найважливіших моментів для менеджера, адже саме цілі визначають успіх або провал проєкту.

2. Формулювання цілей і завдань

2.1. Цілі проєкту: визначення та значення

Після того як проєкт отримує офіційний старт, наступний етап – чітке визначення того, що саме потрібно досягти. Формулювання цілей – один із найважливіших моментів для менеджера, адже саме цілі визначають успіх або провал проєкту.

Ціль – це бажаний кінцевий стан, до якого прагне команда. Вона повинна бути зрозумілою всім учасникам, легко інтерпретованою та спрямованою на створення конкретної цінності.

Хороша ціль ніколи не формулюється як «зробити проєкт успішно». Вона має бути конкретною: наприклад, «створити веб-портал для клієнтів, який дозволить автоматизувати замовлення і зменшити навантаження на операторів на 30%».

Цілі часто формують за підходом SMART:

- S – Specific (конкретна)
- M – Measurable (вимірювана)
- A – Achievable (досяжна)
- R – Relevant (релевантна бізнесу)
- T – Time-bound (обмежена в часі)

Приклад SMART-цілі: «Розробити мобільний застосунок для онлайн-замовлення доставки, який дозволить скоротити час обробки одного замовлення до 30 секунд протягом трьох місяців.»

2.2. Завдання проєкту: структурні кроки до цілі

Якщо ціль – це пункт призначення, то завдання – це «дорога» до нього. Завдання повинні бути логічно пов’язані з ціллю, реалістичними та такими, що можна відстежувати.

Наприклад, для цілі створення мобільного застосунку завданнями можуть бути:

- провести інтерв’ю з користувачами;
- визначити функціональні вимоги;
- створити прототип;
- розробити бекенд;

- провести тестування;
- запустити MVP.

Хоч ми мінімізуємо нумерацію, завдання завжди формують структурно, адже це робочі елементи.

2.3. Зв'язок між цілями, завданнями та результатами

Зручно уявити цю логіку у вигляді міні-схеми:

```
Бізнес-потреба → Ціль проєкту → Завдання → Артефакти → Результат → Цінність
```

Ця схема показує, що кожен елемент має бути пов'язаний з попереднім. Якщо завдання не веде до цілі – це «зайва діяльність», яка знижує ефективність.

2.4. Типові помилки у формулюванні цілей і завдань

Найпоширеніші проблеми:

- надто загальні цілі («покращити сервіс»);
- неможливість виміряти досягнення;
- відсутність часових рамок;
- завдання не відповідають цілі;
- цілі не прив'язані до реальної бізнес-потреби.

Життєвий приклад: людина хоче «схуднути». Без конкретики це не працює. Але якщо сформувати SMART-ціль («зменшити вагу на 5 кг за 6 тижнів»), процес стає зрозумілим і керованим. Так само й у проєктах.

3. Зацікавлені сторони

Під **зацікавленими сторонами** розуміють усіх осіб або групи, які прямо або опосередковано впливають на проєкт або відчувають на собі його результати. Це можуть бути як внутрішні учасники (менеджери, команда розробки, керівництво компанії), так і зовнішні (замовники, кінцеві користувачі, партнери, постачальники).

Розуміння зацікавлених сторін допомагає уникнути конфліктів, правильно формувати вимоги та забезпечити підтримку проєкту на всіх етапах.

У реальному ІТ-проєкті інтереси сторін можуть суттєво відрізнятися. Замовник бажає отримати продукт у найкоротші строки, користувачам потрібен зручний інтерфейс, а команда розробки хоче мати реалістичні дедлайни та якісні вимоги. Керівництво компанії очікує економічної вигоди, а технічні партнери – стабільних інтеграцій.

Проєктний менеджер має знайти баланс між цими інтересами. Це схоже на роботу диригента: кожен музикант грає свою партію, але лише узгоджене звучання створює композицію.

Найчастішою причиною провалу IT-проєктів є не технології, а комунікація. Якщо не враховано інтереси однієї важливої сторони – проєкт може зіткнутися з опором або непередбачуваними блокерами. Регулярні зустрічі, демонстрації, звіти, прозорі канали зв'язку – це те, що утримує проєкт у рівновазі.

4. Паспорт проєкту: складові та значення

4.1. Сутність та роль паспорта проєкту

Паспорт проєкту – це офіційний документ, що формалізує рішення про початок проєкту, описує його ключові параметри й надає менеджеру повноваження для управління. Він є своєрідною «конституцією» проєкту – базовим документом, який дає відповідь на головні питання: чому проєкт важливий, чого він має досягти, які ресурси потрібні, хто за що відповідає і які критерії успіху.

Оскільки IT-проєкти часто пов'язані з ризиками, невизначеністю та змінами вимог, паспорт дозволяє створити точку фіксації, з якої команда стартує. Це допомагає уникнути розбіжностей у розумінні та робить проєкт керованим.

Паспорт – це також документ, який захищає як команду, так і замовника: він фіксує реальні очікування, обмеження й домовленості на момент старту.

4.2. Основні складові паспорта проєкту

Хоча конкретний формат може відрізнятися в різних компаніях, структура паспорта проєкту зазвичай містить кілька основних блоків. Їх можна розглядати як «фундамент», на якому будуватиметься весь подальший менеджмент.

Назва та ідентифікація проєкту. Назва повинна бути короткою, зрозумілою і відображати суть. Іноді додається код, версія або внутрішній ідентифікатор.

Обґрунтування (Justification). Це відповідь на питання: чому цей проєкт взагалі варто робити? Тут описується бізнес-потреба, проблема або можливість. У випадку IT це може бути потреба у модернізації системи, підвищенні ефективності, виході на новий ринок, скороченні витрат тощо.

Мета проєкту. Мета формулюється чітко і вимірювано. Це те, що команда має отримати в результаті реалізації. Наприклад: «створити мобільний застосунок для замовлення доставки, який дозволить збільшити кількість клієнтів на 20% протягом року».

Завдання та обсяг робіт (Scope). Тут визначається, що входить у проєкт, а що не входить. Це важливий блок, адже нечітко визначений обсяг робіт – одна з головних причин проблем у IT-проєктах. Вказуються основні функції продукту, модулі, процеси, які мають бути реалізовані.

Зацікавлені сторони (Stakeholders). Перелік осіб і груп, які впливають або зазнають впливу від проєкту. Для кожної сторони може бути визначено роль, рівень впливу, очікування.

Ролі та відповідальність. Визначаються ключові ролі: керівник проєкту, власник продукту, технічний лідер, команда розробників, експерти тощо. Зазначається, хто за що відповідає, хто приймає рішення й хто є джерелом інформації.

Строки реалізації. Вказуються ключові дати: старт, орієнтовне завершення, етапи (milestones). У водоспадних проєктах строки зазвичай фіксовані, у Agile – орієнтовні.

Ресурси та бюджет. Позначається, які ресурси потрібні: людські, технічні, програмні. Може включати оцінку бюджету або його діапазон.

Ризики та обмеження. У паспорт включають ключові ризики, які вже відомі на старті (наприклад, залежність від сторонніх інтеграцій, нестабільність вимог). Обмеження включають речі, які не можна змінити: законодавчі вимоги, бюджетні рамки, регламент компанії.

Критерії успіху. Це чіткі показники, за якими визначатимуть, чи досяг проєкт своєї мети. Наприклад: «система витримує 10 тисяч одночасних користувачів», «запуск здійснено до 1 грудня», «час обробки операції скорочено до 2 секунд».

Розділ	Приклад
Назва проєкту	"CRM+ Web App"
Мета	Автоматизація процесу роботи з клієнтами
Обґрунтування	Підвищення ефективності відділу продажів на 30%
Обсяг	Модуль контактів, аналітика, інтеграція з телефоном
Ролі	PM – Іванова Т.; Product Owner – Сидоренко О.
Зацікавлені сторони	Бізнес, продажі, техвідділ
Ризики	Зміна пріоритетів бізнесу, нестача даних
Обмеження	Бюджет до 200 тис. грн, термін 4 місяці

Таблиця 2.1.4. Приклад простої структури паспорта проєкту.

Спрощений приклад наведено у таблиці 2.1.4. У реальних ІТ-проєктах паспорт може містити десятки сторінок, але принципи лишаються однаковими.

4.3. Значення паспорта проєкту в реальній роботі

Паспорт проєкту має практичне значення, яке важко переоцінити, він:

- **Забезпечує спільне бачення.** У проєкті беруть участь люди з різним досвідом і ролями, тому документ слугує єдиним джерелом істини.
- **Зменшує ризик неправильних очікувань.** Замовник, команда і керівництво розуміють, що саме вони отримають.
- **Надає менеджеру повноваження.** Офіційний документ підтверджує, що проєкт схвалено.
- **Підтримує контроль та прозорість.** Допомагає відстежувати зміни й аргументувати рішення.
- **Захищає команду.** Якщо згодом хтось намагається "розширити обсяг робіт", паспорт стає доказом початкових домовленостей.

Паспорт проєкту можна порівняти з технічним завданням при ремонтних роботах: якщо є затверджений план, усі розуміють, що й у якій послідовності робиться; без плану – завжди будуть суперечки та плутанина.

Хоч паспорт створюється на етапі ініціації, у багатьох ІТ-проєктах він не залишається статичним документом. У гнучких методологіях його можуть переглядати, уточнювати або розширювати. Головне – зберігати контрольованість змін.

Стабільний, але гнучкий паспорт – це компроміс між потребою в структурі і потребою в адаптації.

5. Критерії успіху. Обґрунтування доцільності

5.1. Сутність критеріїв успіху

Критерії успіху – це чіткі ознаки того, що проєкт досягнув своєї мети та приніс очікувану цінність. У різних проєктів успіх виглядає по-різному. Для одного важливо закінчити в строк, для іншого – отримати продукт, яким реально користуються, для третього – отримати економічний результат.

Традиційно успіх оцінюється за «залізним трикутником»(таблиця 2.1.5): Однак сучасне управління ІТ-проєктами виходить за межі цього підходу.

Показник	Суть
Час	Чи завершено проєкт у запланований термін

Бюджет	Чи вписався проєкт у фінансові рамки
Якість	Чи відповідає результат вимогам

Таблиця 2.1.5. Залізний трикутник успіху.

Сьогодні успішний проєкт – це не лише той, що завершено без перевищення бюджету. Успіх – це користь для бізнесу та користувача.

Наприклад, мобільний застосунок може бути розроблений вчасно, але якщо користувачі не будуть ним користуватися, це не успіх.

Цінність може виражатися у:

- ➢ збільшенні продажів,
- ➢ зменшенні витрат,
- ➢ підвищенні задоволення клієнтів,
- ➢ покращенні ефективності внутрішніх процесів.

5.2 Обґрунтування доцільності проєкту

Обґрунтування доцільності (business justification) – це відповідь на питання: «Навіщо ми запускаємо цей проєкт і яку вигоду отримаємо?». Це стратегічна оцінка, яка визначає, чи варто інвестувати ресурси в розробку певного IT-продукту.

У більшості компаній доцільність визначається за такими параметрами:

- проблема або можливість: що саме ми хочемо вирішити;
- очікувана вигода: економія, прибуток, конкурентна перевага;
- ризики (технологічні, фінансові, ринкові);
- альтернативи (купити готовий продукт, відкласти розробку, змінити технологію);
- попередній розрахунок бюджету.

Це наче планування поїздки: перед тим як вирушити, ми оцінюємо маршрут, витрати, ризики, можливі альтернативи й очікувану користь.

5.3. Роль обґрунтування у прийнятті рішень

Якісний документ доцільності допомагає керівництву ухвалювати зважені рішення. Він також є «страховкою» для менеджера: якщо проєкт запущено на основі продуманого аналізу, менше шансів, що в майбутньому його звинуватять у неправильному виборі.

Питання для самоперевірки

1. Що таке стадія ініціації проєкту?
2. Чому ініціація є важливою для подальшого успіху проєкту?
3. Які ключові завдання виконуються під час ініціації?

4. Що означає формулювання цілей проєкту?
5. Чим цілі відрізняються від завдань?
6. Якими мають бути правильно сформульовані цілі (за якими критеріями)?
7. Хто такі зацікавлені сторони проєкту?
8. Чому важливо визначати зацікавлені сторони на ранньому етапі?
9. Які приклади зацікавлених сторін можна назвати?
10. Що таке паспорт проєкту?
11. Які основні складові паспорта проєкту?
12. Для чого менеджеру потрібен паспорт проєкту?
13. Що включає обґрунтування доцільності проєкту?
14. Чому важливо оцінювати доцільність перед запуском проєкту?
15. Що таке критерії успіху проєкту?
16. Як критерії успіху допомагають контролювати виконання проєкту?
17. Чому етап ініціації впливає на якість планування?
18. Які ризики можуть виникнути, якщо етап ініціації виконаний недостатньо добре?

ТЕМА 2.2
ПЛАНУВАННЯ ПРОЄКТУ. ІЄРАРХІЧНА СТРУКТУРА РОБІТ (WBS)

План

1. Планування як процес.
2. Ієрархічна структура робіт (WBS).
3. Формування завдань. Оцінка тривалості.
4. Розподіл ресурсів.
5. Побудова календарного плану. Інструменти планування.

Ключові слова: планування, структура робіт, WBS, завдання, ресурси, календар, графік, етапи, оцінка, інструменти

Key words: planning, work breakdown structure, WBS, tasks, resources, calendar, schedule, stages, estimation, tools

1. Планування як процес

Планування – це сукупність взаємопов'язаних дій, через які менеджер та команда визначають, що саме потрібно зробити, яким чином це буде виконано, хто відповідатиме за конкретні частини роботи та в які строки відбудеться реалізація.

На практиці планування включає визначення обсягу проєкту, структурування робіт, оцінку строків, аналіз ризиків, вибір інструментів та методології. Процес планування дуже схожий на підготовку до подорожі. Якщо людина збирається в дорогу без плану – вона ризикує забути важливі речі, потрапити не в той маршрут, витратити більше часу й коштів. Але коли складається маршрут, оцінюється бюджет, визначаються критичні точки – подорож проходить набагато передбачуваніше.

У ІТ такий процес є навіть важливішим, оскільки проєкти часто включають в себе багато залежностей: технічних, кадрових, бізнесових. Уміння правильно сформувати план дозволяє команді працювати системно й не втрачати фокус.

Важливо підкреслити, що план – це гнучкий документ. У гнучких методологіях планування відбувається на кількох рівнях: спочатку загальне бачення, потім – більш детальне розбиття задач у кожній ітерації. У традиційних підходах планування є детальним на старті, однак його теж можуть коригувати при появі нових умов.

Основною метою планування є не створення красивої документації, а узгодження між усіма учасниками, що, чому і в які терміни буде зроблено. План – це спільне бачення, яке дозволяє уникати помилок і непорозумінь у майбутньому.

Планування є мостом між ідеєю та виконанням. На стадії ініціації визначаються загальні параметри проєкту, але саме планування перетворює їх у конкретні задачі. У житті це можна порівняти зі зведенням будинку: на початку є лише концепція, але на стадії планування створюються креслення, визначаються етапи та ресурси.

У ІТ-проєктах планування включає:

- ➢ визначення вимог та їх уточнення;
- ➢ розробку структури робіт;
- ➢ визначення пріоритетів;
- ➢ формування графіків;
- ➢ визначення потенційних ризиків;
- ➢ оцінку трудомісткості.

Ці елементи разом формують ясну картину, за якою працює вся команда.

2. Ієрархічна структура робіт (WBS)

2.1. Ієрархічна структура робіт (WBS), принцип її побудови

Одним із найважливіших інструментів планування в управлінні проєктами є **Work Breakdown Structure (WBS)** – ієрархічна структура робіт. Це спосіб розбити великий проєкт на логічні, зрозумілі та керовані частини.

WBS – це систематизована декомпозиція проєкту на елементи, які можна оцінювати, планувати, призначати та контролювати.

WBS допомагає перетворити складний проєкт, що може виглядати як одна велика «хмара задач», у чітку ієрархію елементів. Це як розкласти складну страву на інгредієнти та етапи приготування – що потрібно купити, що потрібно нарізати, що приготувати та за якою черговістю.

WBS створюється зверху вниз: спочатку визначають загальний результат, після чого він розбивається на великі блоки, потім на менші підзадачі, доки кожен елемент не стане достатньо конкретним. Ієрархія дозволяє команді бачити загальну картину і водночас працювати з деталями.

Умовна схема побудови WBS:

```
Проєкт
 ├── Модуль A
 │    ├── Функція A1
 │    ├── Функція A2
 │    └── Тестування модуля
 ├── Модуль B
 │    ├── Функція B1
 │    └── Інтеграційні роботи
 └── Документація
      ├── Керівництво користувача
      └── Технічна документація
```

Така структура дозволяє побачити, із чого саме складається продукт і які частини можуть бути розподілені між командою.

WBS дає змогу:

– отримати чітке уявлення про обсяг робіт;
– уникати «сліпих зон» у плануванні;
– оцінювати час і ресурси точніше;
– формувати календарі та графіки;
– визначати відповідальних;
– спрощувати комунікацію в команді.

Крім того, WBS є основою для інших інструментів управління, наприклад, для діаграм Ганта, оцінки тривалості, матриць відповідальності.

2.2. Приклад застосування WBS у IT-проєкті

Уявімо команду, яка створює мобільний додаток для онлайн-оплат. Без WBS це буде абстрактний набір задач: «зробити додаток». Але при побудові WBS проєкт розподіляється на модулі: авторизація, профіль, оплати, історія транзакцій, інтеграції з банком, тестування, UI/UX, реліз. Кожен модуль має свої підзадачі. Таким чином команда бачить чітку структуру, може розподілити роботу, оцінити навантаження та зрозуміти залежності.

3. Формування завдань. Оцінка тривалості

3.1. Формування завдань: сутність і важливість

Формування завдань – це процес перетворення загальних елементів WBS у конкретні робочі одиниці, зрозумілі розробникам, дизайнерам, тестувальникам та іншим учасникам команди. Завдання мають бути не лише логічними, але й добре сформульованими, щоб виключити різночитання й неправильне виконання.

Хороше завдання в IT має бути:

- конкретним – чітко вказувати, що потрібно зробити;
- вимірюваним – мати критерії завершення (Definition of Done);
- досяжним – відповідати компетенціям виконавців;
- умісним у спринт або ітерацію;
- без двозначностей.

Фактично, ми очікуємо, що завдання буде не просто фрагментом роботи, а одиницею цінності для всієї системи.

Приклад:

Неправильне формулювання: “Зробити авторизацію.”. Таке завдання містить десятки підзадач, і кожен розробник зрозуміє його по-своєму.

Правильне формулювання: “Реалізувати форму логіну з валідацією email/password, інтеграцією з API, відображенням помилок та записом токена в локальне сховище.”. Таке завдання має чіткий результат і зрозумілі межі.

3.2. Оцінка тривалості: логіка та підходи

Оцінка – це спроба передбачити, скільки часу та зусиль потребує виконання конкретного завдання. В IT оцінювання складне через невизначеність, залежності між компонентами, непередбачувані технічні труднощі. Водночас правильна оцінка є основою планування.

Основні підходи до оцінки:

- Експертна оцінка. Команда або окремі спеціалісти оцінюють завдання, спираючись на свій досвід. Це найпоширеніший спосіб, особливо в Agile.
- Оцінка за аналогами. Оцінювання здійснюється шляхом порівняння з подібними завданнями, що виконувалися раніше.
- Тривалість + ризики. Час оцінюється як сума: оптимістичний сценарій + реалістичний + песимістичний, що дає середню оцінку за методом PERT.
- Story Points (у Scrum). Тут оцінюється не час, а складність, залежності та невизначеність. Однак тривалість усе одно виводиться на основі продуктивності команди (velocity).

3.3. Фактори, що впливають на точність оцінки

У реальних IT-проєктах оцінки часто «плавають». Причини:

- неповні або нечіткі вимоги;
- технічна невизначеність;
- зовнішні залежності: API, сторонні сервіси;
- нестача досвіду у виконавця;

- неправильний розподіл складних задач (великі задачі завжди оцінюються неточно).

Тому важливо **декомпозувати завдання до дрібних частин** – це підвищує точність оцінки в рази.

Підхід	Коли добре працює	Мінуси
Експертна оцінка	Швидкі задачі, досвідчені команди	Суб'єктивність
Оцінка за аналогами	Повторювані задачі	Немає аналогів – немає оцінки
PERT	Висока невизначеність	Потребує багато часу
Story Points	Agile-команди, Scrum	Початківці плутаються, потрібен досвід

Таблиця 2.2.3. Порівняння видів оцінок у IT.

4. Розподіл ресурсів

4.1. Розподіл ресурсів: люди, час, інструменти

Після оцінки тривалості менеджер переходить до розподілу ресурсів. Ресурси – це не лише команда, а й час, обладнання, інфраструктура, програмні інструменти й навіть увага.

Управління ресурсами включає:

- визначення доступності членів команди;
- розподіл задач так, щоб уникнути перевантаження;
- врахування робочих графіків, відпусток, форс-мажорів;
- узгодження пріоритетів із замовником.

У класичному управлінні проєктами існує поняття "трикутник обмежень":

```
 Якість
   │
   │
Час ——— Вартість
```

Змінюючи один елемент, ми впливаємо на інші. Якщо зменшити час – зросте вартість або впаде якість. Якщо зменшити команду – зросте тривалість. Це фундаментальна логіка проєктного менеджменту.

Види ресурсів у IT-проєкті:

- Людські ресурси: розробники, тестувальники, DevOps-інженери, аналітики, дизайнери.
- Технічні ресурси: сервери, хмарні платформи, ліцензії, тестове обладнання.
- Адміністративні ресурси: менеджмент, закупівлі, логістика.
- Інформаційні ресурси: документація, дані, специфікації.

Менеджер має враховувати кожен тип, щоб уникнути “вузьких місць”.

Приклади як неправильний розподіл ресурсів руйнує проєкт:

Приклад 1. Один розробник на все

Компанія виділяє одного програміста на мобільний застосунок, який потребує backend, UI, інтеграцій. Він не встигає, терміни зриваються, якість падає. Висновок: неправильний розподіл ресурсів = неправильні очікування.

Приклад 2. Немає тестувальника

Розробники тестують самі себе. Помилки проскакують у релізи. Клієнт невдоволений. Висновок: економія на ролях призводить до втрат у якості.

4.2. Використання інструментів планування ресурсів

У сучасному IT рідко працюють без інструментів. Найпоширеніші:

- Jira – для задач, оцінок і навантаження команди;
- MS Project – для класичного планування (діаграми Ганта);
- Miro – для візуального планування;
- ClickUp, Asana, Trello – для гнучких команд;
- Excel/Sheets – для бюджетів і матриць навантаження.

Використання інструментів підвищує прозорість, контроль і передбачуваність процесів.

5. Побудова календарного плану. Інструменти планування

5.1. Сутність календарного плану

Календарний план (Schedule) – це графічне або табличне відображення робіт проєкту у часі, що демонструє початок, завершення, тривалість і взаємозалежність завдань. Він поєднує в собі результати попередніх кроків планування: сформовану WBS, оцінки тривалості робіт, наявність ресурсів, послідовність залежностей та визначені віхи (milestones).

У календарному плані важливе не лише те, що потрібно зробити, а й коли саме, у якій послідовності, якою командою і з якою інтенсивністю. У цьому сенсі календарний план – це інтегратора всіх попередніх планових документів.

5.2. Логіка побудови календарного плану

Побудова календарного плану завжди починається з розуміння трьох ключових елементів:

- Ієрархічна структура робіт (WBS) – визначає склад проєкту.
- Роботи й оцінки тривалості – визначають обсяг і часові рамки.
- Ресурси та залежності – визначають можливість виконання робіт у конкретні моменти.

На основі цих елементів створюється логічна послідовність, де визначається які роботи є критичними, які мають залежності, які можуть виконуватися паралельно, де можуть бути затримки і наскільки вони допустимі.

Потім ця послідовність переноситься у часову площину – фактично перетворюється на календар.

5.3. Основні методи календарного планування

У практиці управління проєктами використовується кілька базових підходів. Найпоширеніші:

Метод критичного шляху (Critical Path Method, CPM). Це один із найважливіших інструментів. Він дає змогу визначити критичний шлях – послідовність робіт, затримка в яких автоматично затримує весь проєкт.

Критичний шлях – це «кістяк» проєкту. Роботи, які не входять до нього, мають певний запас часу (float), а критичні завжди мають нульовий запас.

Схематично це виглядає так:

```
A → B → C → D (критичний шлях)
  ↘ E ↗
```

Діаграма Ганта. Діаграма Ганта – це візуальна форма календарного плану, яка показує роботи як горизонтальні смуги на часовій шкалі. Вона дозволяє швидко оцінювати тривалість, завантаженість, послідовність і прогрес.

Робота	Тривалість	Графік
Аналіз	5 днів	[redacted]
Розробка	10 днів	[redacted]
Тестування	4 дні	[redacted]

Таблиця 2.2.5. Приклад діаграми Ганта у спрощеному вигляді.

PERT-аналіз. PERT використовується тоді, коли невизначеність тривалості робіт висока. Метод застосовує три оцінки: оптимістичну,

найімовірнішу та песимістичну, що дозволяє розрахувати статистично обґрунтовану тривалість.

5.4. Практичні аспекти побудови календарного плану в IT

IT-проєкти мають свою специфіку. Наприклад:

- Частина робіт виконується паралельно (Back-end та Front-end).
- Роботи можуть залежати від зовнішніх інтеграцій, API, сторонніх команд.
- Команда працює ітеративно, а не лінійно, особливо у Scrum.
- Можливі швидкі зміни вимог, що спотворюють початкові оцінки.

Тому календарне планування в IT завжди включає:

- Гнучкість – план завжди оновлюється.
- Буфери часу – резерви для непередбачених затримок.
- Короткі цикли – поєднання довгострокового плану з короткостроковими спринтами.
- Прогнозування ризиків – аналіз залежностей, які можуть «завалити» графік.

Приклад. Якщо команда створює мобільний застосунок, то дизайн і бекенд можуть працювати паралельно, але інтеграція залежить від готовності API. Тому завдання розміщуються так, щоб інтеграція не блокувала критичний шлях.

5.5. Інструменти календарного планування

Сучасні проєкти використовують як класичні професійні системи, так і прості онлайн-дошки. Інструменти можна поділити на кілька груп.

- **Потужні системи управління проєктами.** Вони дозволяють будувати діаграми Ганта, критичний шлях, планувати ресурси, відстежувати прогрес. (Microsoft Project, Primavera P6, GanttPRO)
- **Інструменти для гнучких команд.** Переважно використовуються у Scrum та Kanban. (Jira, Trello, YouTrack, Monday.com)
- **Комбіновані інструменти.** Можуть містити Гант, backlog, ресурси, KPI. (ClickUp, Notion, Asana)
- **Табличні інструменти.** Використовуються для невеликих команд або простих планів. (Google Sheets, Excel)

5.6. Розуміння календарного плану як «обіцянки команди»

Календарний план – це не просто графік. Це зобов’язання, прийняте командою, менеджером і зацікавленими сторонами. Він допомагає організувати роботу, узгодити очікування, прогнозувати результати, визначати критичні ризики, контролювати виконання.

Якщо команда постійно не вкладається у календарний план, проблема не лише в плані – зазвичай це знак неправильних оцінок, перевантаження або відсутності стабільних процесів.

Питання для самоперевірки

1. Що означає планування у контексті управління проєктами?
2. Чому планування вважається одним із ключових процесів у проєкті?
3. Що таке ієрархічна структура робіт (WBS)?
4. Для чого використовують WBS у проєкті?
5. Яким чином WBS допомагає структурувати роботи?
6. Що таке робочий пакет (work package)?
7. Як визначити межі робочого пакета?
8. Чому важливо правильно формулювати завдання у проєкті?
9. Які методи використовують для оцінки тривалості робіт?
10. Що таке критичний шлях у проєкті?
11. Навіщо здійснюється розподіл ресурсів?
12. Які ресурси можуть бути необхідними для виконання робіт?
13. Чому неправильний розподіл ресурсів може призвести до проблем у проєкті?
14. Що таке календарний план проєкту?
15. У чому відмінність між діаграмою Ганта та мережевим графом?
16. Які інструменти найчастіше використовують для планування проєктів?
17. Як планування допомагає контролювати прогрес виконання?
18. Чому важливо регулярно оновлювати календарний план?

ТЕМА 2.3
УПРАВЛІННЯ ЗМІНАМИ ТА КОНФЛІКТАМИ

План

1. Природа змін у проєкті.
2. Управління змінами. Журнал змін.
3. Конфлікти у командах. Типи конфліктів.
4. Методи врегулювання конфліктів.
5. Роль керівника у вирішенні конфліктів.

Ключові слова: зміни, управління, конфлікт, команда, врегулювання, журнал, стратегія, переговори, ризик, комунікація

Key words: changes, management, conflict, team, resolution, log, strategy, negotiation, risk, communication

1. Природа змін у проєкті

Зміни – це природна частина будь-якого проєкту. Вони виникають не тому, що хтось неправильно спланував роботу, а тому, що проєкт функціонує в живому середовищі. Зміни – це не помилка, а нормальний процес еволюції проєкту.

Причини змін можна умовно поділити на кілька груп, хоча на практиці вони взаємопов’язані. Найчастіше зміни спричиняють:

- розвиток бізнесу та перегляд бізнес-цілей;
- конкретизація або переосмислення вимог замовником;
- технічні відкриття і нові технологічні можливості;
- реакція на ринок, конкурентів або поведінку користувачів;
- виявлення технічних ризиків і проблем;
- юридичні, регуляторні або безпекові вимоги.

В ІТ-проєктах зміни є настільки частими, що сучасні методології фактично будуються навколо здатності швидко адаптуватися. Саме тому Agile підходи наголошують, що готовність до змін важливіша за слідування початковому плану.

У житті ми теж постійно адаптуємо плани. Уявімо ремонт у квартирі: вже після монтажу перегородок власник може вирішити розширити кухню або перемістити розетки. Це неприємно, але абсолютно природно, бо нове бачення часто з’являється лише після того, як людина побачила результат на власні очі.

Так само і в проєктах: розуміння цінності продукту розвивається по мірі його створення.

Попри природність змін, вони несуть ризики: зрив строків, збільшення бюджету, перегруз команди, технічні переробки, конфлікти між замовником і виконавцем. Неконтрольовані зміни можуть призвести до явища **scope creep** – «повзучого розширення вимог», коли проєкт збільшується до нескінченності, не просуваючись до завершення.

Тому ключовим завданням менеджера є не запобігти змінам повністю, а запровадити механізм управління ними. Зміни повинні бути проаналізовані, оцінені, погоджені, документально зафіксовані та впроваджені контрольовано.

2. Управління змінами. Журнал змін

2.1. Управління змінами: сутність процесу

Управління змінами – це системна діяльність із прийому, оцінювання, погодження, впровадження та документування змін у проєкті. Це процес, який дозволяє перетворити хаотичні ініціативи на прогнозований і контрольований механізм.

У класичному підході управління змінами можна описати як послідовність:

1) Зміна пропонується (зазвичай від замовника або команди).
2) Зміна аналізується менеджером і технічною командою: вплив на строки, бюджет, ризики, обсяг робіт.
3) Зміна погоджується або відхиляється.
4) Якщо погоджена – включається до плану.
5) Усі дії фіксуються у спеціальному документі – журналі змін.

Цей процес дає змогу зберегти прозорість і уникнути типових конфліктів, коли замовник каже: «Ми ж домовлялися зробити ось це!», а команда відповідає: «Це не було в початковому ТЗ». Документовані зміни – це спільна пам’ять проєкту.

2.2. Інструменти управління змінами

Методи можуть відрізнятися залежно від вибраної методології, але загальна логіка схожа. У Waterfall зміни контролюються дуже суворо, через формальні процедури. В Agile зміни є частиною процесу: їх обробляють на рівні беклогу, пріоритизації й інкрементної розробки.

Проте в обох випадках використовується документ під назвою журнал змін.

2.3. Журнал змін: призначення та структура

Журнал змін (Change Log) – це офіційний запис усіх запропонованих, погоджених і відхилених змін у проєкті, що дозволяє відстежувати історію рішень. Він допомагає побачити, як еволюціонував продукт, хто ініціював зміну, коли вона була погоджена та який вплив мала.

Журнал змін – це інструмент прозорості й захисту як для команди, так і для замовника. У ньому можна легко встановити, які зміни призвели до збільшення бюджету або які рішення вплинули на строки.

Для зручності журнал часто ведеться у вигляді таблиці (приклад наведено у таблиці 2.3.2):

№ зміни	Дата	Опис змін	Ініціатор	Вплив (строки/бюджет/якість)	Рішення	Коментар
01	12.03	Додати нову функцію експорту	Замовник	+ 2 тижні	Погоджено	Включено в реліз 2
02	15.03	Заміна технології фронтенду	Команда	Збільшення ризиків	Відхилено	Нестабільно для проєкту

Таблиця 2.3.2. Пликлад журналу змін.

Це проста, але надзвичайно сильна схема контролю.

2.4. Управління змінами в Agile-командах

У гнучких методологіях зміни не сприймаються як проблема – навпаки, вони є частиною процесу. У Scrum усі зміни відображаються у беклозі продукту, де Product Owner керує пріоритетами. Зміна може бути додана до беклогу в будь-який момент, але не може увірватися в поточний спринт без погодження.

У Kanban зміни ще простіше: картки додаються, переміщуються або уточнюються без фіксованих ітерацій, але обмеження на роботу (WIP-ліміти) не дозволяють команді «захлинутися».

2.5. Баланс між гнучкістю та контролем

Зміни – це можливість зробити продукт кращим, але лише якщо ними керувати правильно. Надмірна жорсткість заважає розвитку продукту. Надмірна гнучкість створює хаос. Завдання менеджера – знайти баланс між стабільністю та адаптивністю.

Найуспішніші IT-команди вміють не лише реагувати на зміни, а й передбачати їх, створюючи запас часу, резерв бюджету та інструменти для швидкої адаптації.

3. Конфлікти у командах. Типи конфліктів

3.1. Поняття конфлікту в IT-команді

Конфлікт у проєкті – це зіткнення інтересів, цінностей, поглядів або потреб учасників, що створює напругу й заважає конструктивній роботі. У команді розробників це може проявитися по-різному: хтось не погоджується з архітектурним рішенням, хтось вважає, що строк занадто короткий, а хтось – що тестування працює надто повільно й блокує реліз.

У здоровому середовищі конфлікти можуть бути навіть корисними – вони стимулюють обговорення, покращують рішення, допомагають виявити проблеми раніше. Але якщо конфлікт ігнорувати або неправильно обробляти, він переходить у фазу деструкції, створює токсичну атмосферу, знижує продуктивність і шкодить проєкту.

3.2. Чому конфлікти виникають у IT-командах

IT – це сфера з високим темпом, неоднозначними вимогами й різними ролями. Розробник бачить продукт з одного боку, тестувальник – з іншого, менеджер – з третього. Також додається фактор особистостей: хтось працює швидко, хтось потрібен більше часу; хтось спілкується прямо, а інший – м’яко та опосередковано. Саме через різність людей та завдань конфлікти є природною частиною роботи.

Приклад із життя: дизайнер створює інтерфейс, який виглядає ефектно, але розробник каже, що його реалізація займе додаткові 3 тижні. Дизайнер обурюється: «Чому так довго?» – і починається дискусія, яка легко може перерости у конфлікт, якщо її не скерувати правильно.

3.3. Типи конфліктів у командах

Щоб правильно реагувати на конфлікт, важливо розуміти, з яким його типом має справу менеджер. Найпоширеніші види такі:

Конфлікт цілей. Виникає, коли члени команди мають різні уявлення про пріоритети. Наприклад, розробник хоче дати більше функцій, а менеджер – доставити мінімально необхідний продукт у строк.

Конфлікт ролей. Ситуація, коли незрозуміло, хто за що відповідає. У IT це трапляється часто: «Це повинен був зробити DevOps, ні – QA, ні – Back-end».

Конфлікт ресурсів. Нестача часу, людей, бюджету чи обладнання. Команди починають «боротися» за ресурси: одна команда потребує середовища для тестування, а друга – для розгортання.

Конфлікт процесів. Стосується способів роботи. Наприклад, одна частина команди підтримує Scrum, інша – Kanban.

Міжособистісний конфлікт. Найскладніший. Він виникає не через процеси, а через темперамент, характер або стиль спілкування. Люди можуть бути однаково професійними, але несумісними у поведінці.

Тип	Суть	Типова ситуація
Цілей	Різні очікування щодо результату	Власник продукту хоче MVP, розробники хочуть «ідеальний продукт»
Ролей	Відсутність чіткої відповідальності	Ніхто не бере задачу, бо «це не моє»
Ресурсів	Недостатність часу/людей/інструментів	Команди «конкурують» за тестове середовище
Процесів	Розбіжності в методах роботи	Scrum vs Kanban
Особистісний	Несумісність стилів поведінки	Хтось занадто прямолінійний, інший – чутливий

Таблиця 2.3.3. Узагальнена таблиця типів конфліктів у командах.

3.4. Емоційні та раціональні рівні конфлікту

Конфлікт завжди має раціональний рівень (змістовний аспект) і емоційний рівень (реакції, образи, ставлення). У технічних командах часто намагаються «вирішити проблему логікою», але якщо ігнорується емоційна складова, конфлікт лише поглиблюється.

Наприклад, тестувальник зауважує, що баги повторюються. Розробник сприймає це як особисте звинувачення. Якщо менеджер не допоможе розвести емоції від фактів, конфлікт стане міжособистісним.

4. Методи врегулювання конфліктів

4.1. Методи врегулювання конфліктів

Управління конфліктами – одна з найважливіших компетенцій менеджера. Нижче описано основні підходи, які застосовуються у IT-командах.

1. Уникнення. Метод працює лише тоді, коли конфлікт незначний або «сам розсмокчеться». Але часто створює ризик накопичення проблем.

2. Пристосування. Одна сторона жертвує своїми інтересами заради миру. Це корисно, коли питання не принципове, але небезпечно, якщо повторюється систематично.

3. Компроміс. Кожна сторона отримує частину бажаного, але не все. Корисний у ситуаціях із ресурсами чи строками.

4. Конкуренція. Одна сторона «перемагає». Цей метод ефективний лише коли потрібно швидко прийняти рішення (криза, реліз, аварія).

5. Співробітництво. Найкращий спосіб у більшості команд. Сторони разом шукають рішення, яке задовольнить усіх. Потребує часу, але будує довіру.

4.2. Практичні інструменти врегулювання конфліктів

Менеджер може застосовувати різні інструменти, наприклад:

- **медіація** – коли менеджер виступає як нейтральний посередник;
- **структуровані обговорення** – обмеження часу, визначення правил: «говорити по черзі», «фокус на проблемі, а не на людині»;
- **переговори по інтересах**, а не по позиціях;
- **визначення границь відповідальності** – документування ролей;
- **ретроспективи**, де команда обговорює проблеми без звинувачень;
- **візуалізація процесів** (Kanban) – часто знімає частину напруги.

4.3. Профілактика конфліктів

Найкращий спосіб зменшити кількість конфліктів – створити умови, у яких вони виникатимуть рідше. Це включає: прозорі процеси, чіткі ролі, чесну комунікацію, повагу до різних стилів роботи, регулярний зворотний зв’язок, психологічно безпечне середовище.

У багатьох ІТ-командах конфлікти зникають після впровадження елементарних практик: щоденних стендапів, ретроспектив, візуалізації задач, чіткого формулювання Definition of Done.

5. Роль керівника у вирішенні конфліктів

5.1. Ролі керівника

1. Керівник як фасилітатор комунікації

Першою і основною роллю керівника у вирішенні конфліктів є роль фасилітатора, тобто людини, яка забезпечує ефективну комунікацію між членами команди. Керівник створює умови, у яких кожен може висловитися, бути почутим, зрозуміти позицію інших та рухатися до рішення.

У рамках проєкту це часто означає організацію зустрічей, медіацію спірних питань, коректне спрямовування дискусій. Керівник має вміти вчасно зупинити емоційне загострення і перенаправити команду з конфлікту на пошук рішень.

Приклад із практики: двоє розробників сперечаються щодо архітектурного підходу. Кожен переконаний, що його варіант кращий. Керівник не повинен обирати «правого», його завдання – допомогти обом зрозуміти критерії прийняття рішення: продуктивність, масштабованість, час розробки, вимоги клієнта. Таким чином конфлікт переводиться з емоційної площини у конструктивну.

2. Керівник як модератор емоцій

У конфліктних ситуаціях важливу роль відіграє емоційний стан людей. Коли емоції беруть верх, логіка та професійність відходять на другий план. Керівник повинен виступати модератором емоцій, допомагаючи команді повернутися до раціонального діалогу.

Це включає такі вміння як знизити напругу, вчасно зробити паузу, підтримати атмосферу взаємоповаги, розпізнати, коли конфлікт стає особистим, а не професійним.

Менеджер не повинен ігнорувати емоції команди, адже прихований емоційний конфлікт рано чи пізно стане явним та болючим. Він має створювати простір, де члени команди можуть відкрито говорити про проблеми, не боячись бути осудженими.

3. Керівник як арбітр і регулятор правил

Хоча в ідеалі конфлікти мають вирішуватися командою самостійно, на практиці керівник часто виступає у ролі арбітра, тобто людини, що приймає остаточне рішення. Це необхідно, коли конфлікт зайшов у глухий кут, і подальше обговорення не веде до результату.

Разом із тим, керівник також є регулятором правил, що підтримують структуру взаємодії. Наявність чітких правил командної роботи (наприклад, правила комунікації, форматів мітингів, відповідальності, ескалації) зменшує кількість конфліктів і полегшує їх вирішення.

Приклад: якщо тестувальники постійно скаржаться, що завдання надходять «у останній момент», керівник має не лише вирішити суперечку між ролями, а й створити правило: «Жодне завдання не може бути передане в тестування без завершення коду, документації та мінімального самотесту».

4. Керівник як наставник і вихователь командної культури

Дуже важлива роль – формування культури, яка запобігає появі руйнівних конфліктів. Культура відкритості, поваги, відповідальності та командності не виникає сама по собі. Саме керівник задає тон у спілкуванні, демонструє поведінку, створює правила взаємодії.

Менеджер навчає команду ефективно давати зворотний зв'язок, аргументувати свою позицію, ставити реалістичні очікування і визнавати

помилки. Хороша культура робить команду стійкою до стресу, а конфлікти – короткими і конструктивними.

5.2. Інструменти, які використовує керівник у роботі з конфліктами

Для вирішення конфліктів керівники застосовують різні інструменти: від неформальних розмов до чітких бюрократичних механізмів. Найпоширеніші з них наведено у таблиці 2.3.5.

Інструмент	Суть і застосування
One-to-one зустрічі	Індивідуальна робота з емоціями, мотивацією, непорозуміннями.
Командні обговорення	Створюють спільне бачення проблеми та шляхів вирішення.
Фасилітація мітингів	Керівник керує дискусією, щоб вона не виходила з-під контролю.
Чіткі правила комунікації	Регламентують взаємодію й знижують кількість конфліктних ситуацій.
Ескалація питання	Винесення складного конфлікту на вищий рівень відповідальності.
Наукові методики управління конфліктами	«win-win», компроміс, співпраця, уникання – залежно від ситуації.

Таблиця 2.3.5. Перелік найпоширеніших інструментів вирішення конфліктів.

Ці інструменти створюють системний підхід до роботи з конфліктами, замість хаотичних та емоційних рішень.

5.3. Вплив керівника на динаміку команди

Конфлікти – це не завжди зло. У багатьох випадках вони можуть бути корисними, якщо спрямовані на розвиток команди та підвищення якості рішень. Завдання керівника – не уникати конфліктів, а керувати ними так, щоб вони ставали точками зростання.

Керівник впливає на «клімат» команди, формує рівень довіри, відкритість і готовність до співпраці. Від його стилю управління залежить, чи будуть конфлікти в команді короткими та продуктивними, чи довгими і деструктивними.

Питання для самоперевірки

1. Що таке зміни у проєкті?
2. Чому зміни є невід’ємною частиною будь-якого проєкту?
3. Які фактори найчастіше спричиняють необхідність змін?
4. Що означає управління змінами?
5. Для чого створюється журнал змін?
6. Які дані зазвичай фіксуються у журналі змін?
7. Хто може ініціювати зміну у проєкті?
8. Які наслідки можуть виникнути у разі неконтрольованих змін?
9. Що таке конфлікт у команді?
10. Які типи конфліктів найчастіше зустрічаються в проєктах?
11. Чому конфлікти іноді є корисними для команди?
12. Які методи врегулювання конфліктів існують?
13. У чому полягає метод компромісу?
14. Що означає метод співпраці у вирішенні конфліктів?
15. Чим небезпечна стратегія уникнення?
16. Яку роль відіграє керівник у вирішенні конфліктів?
17. Чому важливо зберігати нейтральність під час посередництва у конфлікті?
18. Як ефективне управління змінами допомагає зменшити кількість конфліктів?

ЗМІСТОВИЙ МОДУЛЬ 3
УПРАВЛІННЯ КЛЮЧОВИМИ АСПЕКТАМИ ІТ-ПРОЄКТІВ

ТЕМА 3.1
УПРАВЛІННЯ ЯКІСТЮ. УПРАВЛІННЯ ЛЮДСЬКИМИ РЕСУРСАМИ. УПРАВЛІННЯ ПОСТАВКАМИ

План

1. Поняття якості проєкту.
2. Методи забезпечення якості.
3. Управління персоналом.
4. Формування команди.
5. Розподіл ролей.
6. Управління постачанням ресурсів і послуг. Контроль постачань.

Ключові слова: якість, ресурси, команда, ролі, персонал, постачання, контроль, стандарти, продуктивність, відповідальність

Key words: quality, resources, team, roles, personnel, supply, control, standards, productivity, responsibility

1. Поняття якості проєкту

1.1. Поняття якості

Під якістю проєкту у сфері ІТ розуміють ступінь відповідності продукту вимогам, очікуванням і стандартам, визначеним під час ініціації та планування. Це не лише правильність роботи програмного забезпечення, а й зручність використання, доступність, безпека, продуктивність, масштабованість, відповідність бізнес-цілям.

Якість в ІТ складається з трьох взаємопов’язаних частин: якість процесу, якість продукту та якість управління.

- Якість процесу – наскільки ефективно організовано розробку: чи є зрозумілі етапи, чи працює комунікація, чи є документація та стандарти.
- Якість продукту – чи виконує програма свої функції без помилок, чи задовольняє потреби користувачів.
- Якість управління – чи контролюються зміни, чи коректно оцінюються ризики, чи своєчасно приймаються рішення.

У реальних проєктах якість починається не з тестування, а з моменту першої розмови з замовником. Якщо неправильно зрозуміти очікування, навіть

ідеально написаний код не буде якісним, бо продукт не виконуватиме потрібних функцій.

1.2. Основні підходи до розуміння якості

Якість можна оцінювати через різні призми. У практиці IT виділяють 4 підходи.

Підхід відповідності вимогам. Найпоширеніший у менеджменті. Якість означає: «робить саме те, що вказано у специфікації». Але його недостатньо, якщо вимоги зібрані неправильно.

Підхід орієнтації на користувача. Якість – це задоволення очікувань кінцевих користувачів. Програма може відповідати специфікації, але бути незручною.

Підхід орієнтації на цінність. Якість визначається тим, наскільки продукт приносить користь бізнесу. Наприклад, простий, але стабільний додаток може давати більше цінності, ніж складний і дорогий.

Підхід процесного мислення. Якість – результат стабільного процесу. Якщо команда працює хаотично, хорошу якість можна отримати лише випадково.

Найкраще працює поєднання підходів, коли увага приділяється і вимогам, і користувачам, і процесу.

2. Методи забезпечення якості

Забезпечення якості (Quality Assurance, QA) – це система превентивних заходів, яка має на меті запобігати появі дефектів ще до їх виникнення. На відміну від контролю якості (Quality Control), що шукає помилки після їх появи, QA формує процеси.

У сучасних IT-проєктах застосовуються різні методи забезпечення якості. Їх можна умовно поділити на методи: процесні, технічні та організаційні.

1. Процесні методи

Процесні методи допомагають будувати передбачуваний, повторюваний та контрольований процес розробки.

- **Стандартизація процесів** – використання описаних процедур, шаблонів, інструкцій.
- **Код-рев'ю** – взаємна перевірка коду розробниками, що дозволяє виявляти помилки на ранньому етапі.
- **Ретроспективи** – регулярні зустрічі для аналізу минулих спринтів та безперервного вдосконалення.
- **Автоматизація** – CI/CD-процеси, автоматизовані збірки, перевірки стилю коду.

Такі методи дають змогу підвищити якість ще до появи продукту, забезпечуючи стабільність процесу.

2. Технічні методи

Технічні методи стосуються безпосередньо створення продукту.

- **Тестування** – функціональне, регресійне, інтеграційне, навантажувальне.
- **Автоматизовані тести** – перевірки, що виконуються автоматично на кожному етапі розробки.
- **Аналіз архітектури** – оцінка архітектурних рішень до початку імплементації.
- **Прототипування** – швидке створення спрощених версій продукту для перевірки ідей.

Наприклад, уявімо, що команда створює мобільний додаток. Прототип дозволяє перевірити логіку інтерфейсу, не витрачаючи місяці на розробку. Автоматизовані тести гарантують, що новий функціонал не зламає вже працюючі частини.

3. Організаційні методи

Це методи, пов’язані з управлінням, комунікацією та взаємодією.

- **Регулярні зустрічі з замовником** – уточнення вимог, пріоритетів.
- **Управління ризиками** – прогнозування проблем і розробка стратегій реагування.
- **Документування** – збереження рішень, вимог, змін.
- **Аналіз зацікавлених сторін** – робота з очікуваннями різних учасників проєкту.

У складних проєктах часто саме відсутність комунікації призводить до помилок – а не технічні проблеми.

3. Управління персоналом

Управління персоналом – це система процесів, спрямованих на залучення, організацію, розвиток, мотивацію та утримання людей, які працюють над проєктом. В IT ця складова набуває особливої ваги через постійну динаміку, необхідність швидко адаптуватися до нових вимог, різноманітні типи ролей та високу конкуренцію за спеціалістів.

Управління персоналом включає планування людських ресурсів, добір спеціалістів, адаптацію нових учасників, розподіл ролей, підтримку мотивації, створення сприятливої атмосфери, вирішення конфліктів, розвиток компетенцій та контроль ефективності.

Управління персоналом в ІТ-проєкті умовно проходить через кілька фаз, які хоча й можуть накладатися одна на одну, але утворюють логічну послідовність:

- **Формування потреби.** Менеджер визначає, яких фахівців потребує проєкт, який обсяг роботи, які компетенції й на який термін.
- **Залучення спеціалістів.** Пошук кандидатів, співбесіди, технічна оцінка, ухвалення рішення.
- **Інтеграція в команду.** Онбординг, введення в контекст проєкту, знайомство з процесами.
- **Організація роботи.** Розподіл ролей, визначення зон відповідальності, планування комунікацій.
- **Розвиток та мотивація.** Навчання, зворотний зв'язок, підтримка професійних потреб.
- **Оцінювання результатів.** Аналіз ефективності, корекція ролей, перерозподіл завдань.

Якщо провести аналогію з життям, управління персоналом нагадує створення музичного гурту: мало знайти талановитих музикантів – потрібно, щоб вони звучали гармонійно, працювали у єдиному ритмі та мали спільну мету.

Успішне управління персоналом неможливе без чіткої та структурованої комунікації. В ІТ-командах, де різні фахівці можуть працювати у різних часових поясах або віддалено, комунікація стає частиною архітектури проєкту.

Менеджер проєкту формує–чіткі канали (наприклад, Slack, Teams), правила для зустрічей (регулярні стендапи, планування, демо), очікування від кожного учасника (чіткі дедлайни, регулярні статуси). Прозорість комунікації – основа довіри та дисципліни.

4. Формування команди

Формування проєктної команди – це не випадкове зібрання фахівців, а цілеспрямована побудова структури, яка зможе реалізувати задуманий продукт. Менеджер має враховувати технічні компетенції, сумісність характерів, здатність до колаборації, рівень самостійності та мотивацію кожного члена команди.

Структура команди може бути різною, але зазвичай включає такі ролі:

- **Менеджер проєкту** – координує роботу, відповідає за строки, ризики та взаємодію.
- **Власник продукту** (якщо Agile) – формує бачення та пріоритети продукту.
- **Розробники** – створюють функціонал.

- **QA-спеціалісти** – забезпечують якість.
- **Бізнес-аналітики** – формалізують вимоги.
- **DevOps-інженери** – забезпечують стабільність інфраструктури.
- **UI/UX дизайнери** – створюють інтерфейси.

Формування команди можна порівняти з набірною мозаїкою: кожен елемент має своє місце, і тільки правильне поєднання створює цілісність.

Технічні та поведінкові фактори під час добору персоналу:

Тип факторів	Приклади	Значення
Технічні компетенції	Мови програмування, фреймворки, досвід з базами даних	Визначають здатність виконати роботу
Поведінкові компетенції	Комунікабельність, відповідальність, стресостійкість	Визначають якість співпраці
Проєктний досвід	Участь у подібних проєктах	Зменшує ризики помилок
Готовність до навчання	Відкритість до нових процесів	Важливо в гнучких методологіях

Таблиця 3.1.4. Типи факторів та їх значення.

4.1. Формування команди та модель Такмана

Для розуміння групової динаміки часто використовують модель Брюса Такмана, яка описує етапи становлення команди:

- **Forming** – знайомство, обережність
- **Storming** – конфлікти, боротьба за вплив
- **Norming** – стабілізація, узгодження правил
- **Performing** – продуктивність, синергія

У контексті IT-проєктів ця модель допомагає менеджеру планувати комунікації, передбачати конфлікти та правильно підтримувати команду на різних етапах.

4.2. Мотивація та розвиток персоналу

Для довготривалих проєктів важливо створити умови, де фахівці можуть рости. Це включає:

- можливість брати участь у рішеннях;
- навчальні курси, конференції;
- внутрішні knowledge-sharing сесії;

- ротацію ролей;
- регулярний зворотний зв’язок.

5. Розподіл ролей

Роль – це функціональна позиція, яка має чіткі завдання, межі відповідальності та результати, що очікуються від її виконавця. Роль не дорівнює посаді: одна людина може виконувати кілька ролей, або навпаки – одна роль може бути розподілена між кількома виконавцями.

Розподіл ролей – це процес визначення, хто за що відповідає, як відбувається взаємодія між членами команди й які очікування покладаються на кожну позицію. Правильний розподіл ролей гарантує, що в команді немає дублювання функцій, провалених зон відповідальності та «розмитих» обов'язків.

Наприклад, у невеликому стартапі одна особа може поєднувати ролі аналітика, тестувальника та менеджера продукту. У великій корпорації кожна роль є окремою спеціалізацією з чіткими процесами.

У більшості проєктів зустрічаються такі ключові ролі:

- **Проєктний менеджер** – відповідає за план, строки, бюджет, комунікацію, ризики та результати.
- **Власник продукту (Product Owner)** – визначає бачення продукту, пріоритети, зміст беклогу.
- **Аналітик** – формує вимоги, описує бізнес-процеси, допомагає команді зрозуміти цілі продукту.
- **Розробники** – створюють функціональність продукту.
- **Тестувальники** – перевіряють якість і відповідність вимогам.
- **UX/UI спеціаліст** – формує досвід користувача та інтерфейс.
- **DevOps / системні інженери** – забезпечують інфраструктуру, розгортання та стабільність.
- **Техпідтримка / служба експлуатації** – супроводжують продукт після релізу.

У різних методологіях акценти можуть змінюватися, але призначення ролей залишається незмінним: хтось повинен відповідати за продукт, хтось – за процес, хтось – за технічну реалізацію та якість.

Коли ролі не визначені, команда стикається з такими труднощами:

- завдання «падають між стільцями»;
- з’являється плутанина щодо відповідальності;
- виникають конфлікти через очікування;
- зростають ризики затримок;
- ніхто не відповідає за ключові рішення.

Прикладом може бути ситуація, коли не зрозуміло, хто відповідальний за комунікацію із замовником. У результаті один член команди обіцяє одне, інший – інше, і замовник втрачає довіру.

6. Управління постачанням ресурсів і послуг. Контроль постачань

6.1. Управління постачанням

Управління постачанням – це процес планування, пошуку, отримання та контролю всіх зовнішніх ресурсів, необхідних для виконання проєкту. Часто IT-проєкт не може бути реалізований виключно внутрішніми силами – потрібне обладнання, сервіси, ліцензії, послуги підрядників, консультації, хмарні ресурси тощо.

Етапи управління постачанням:

1) Планування потреб. Менеджер визначає, які ресурси потрібні, у якій кількості та коли. Наприклад, сервери для розгортання тестового середовища або підписка на інструмент розробки.

2) Вибір постачальників. Проводиться аналіз ринку, порівняння пропозицій, іноді тендер. Оцінюються ціна, якість, надійність, SLA.

3) Закупівля та договір. Визначаються умови постачання, строки, гарантії, обслуговування.

4) Контроль поставок. Менеджер контролює виконання домовленостей: строки, обсяг, якість, відповідність договору.

5. Приймання та інтеграція ресурсів у проєкт. Ресурс має бути не просто отриманий, а й включений у робочий процес.

6.2. Типи постачань у IT-проєкті

До категорій постачань часто належать:

- **Технічні ресурси:** сервери, хмарні потужності, мережеве обладнання.
- **Програмне забезпечення:** ліцензії, інструменти, фреймворки, доступи.
- **Послуги:** аутсорс-розробка, дизайн, аудит безпеки, консалтинг.
- **Матеріали:** техніка для робочих місць команди.

6.3. Схема контролю постачань

Контроль постачання складається з постійного моніторингу (таблиця 3.1.6.)

Елемент контролю	Що перевіряється?

Строки	Чи прибуло вчасно?
Кількість	Чи відповідає замовленому?
Якість	Чи придатне до використання?
Відповідальність	Хто відповідає за інтеграцію?
Ризики	Що робити, якщо постачання затримується?

Таблиця 3.1.6. Елементи контролю.

Якщо постачання затримується або постачальник не виконує SLA – менеджер має оперативно оцінити ризики, залучити альтернативи або переглянути план робіт.

Розподіл ролей і постачання тісно пов’язані: хтось відповідає за вимоги, хтось – за закупівлю, хтось – за технічну інтеграцію. Без чітких ролей постачання стає хаотичним, а без належного постачання команда не може виконувати свої ролі.

Питання для самоперевірки

1. Що таке якість проєкту?
2. Чому якість є важливим показником успішності проєкту?
3. Які методи забезпечення якості найчастіше використовуються?
4. У чому різниця між контролем якості та забезпеченням якості?
5. Що таке стандарти якості?
6. Чому важливо створювати план управління якістю?
7. Що включає управління персоналом у проєкті?
8. Які навички необхідні для ефективної роботи менеджера персоналу?
9. Чим формування команди відрізняється від набору персоналу?
10. Які фактори впливають на ефективність команди?
11. Для чого проводять розподіл ролей у команді?
12. Які основні ролі можуть бути у проєктній команді?
13. Що таке управління постачаннями в проєкті?
14. Які ресурси можуть потребувати постачання?
15. Для чого створюється план постачань?
16. У чому полягає контроль постачань?
17. Які ризики можуть виникнути при неналежному управлінні постачаннями?

18. Чому узгодженість між командою та постачальниками важлива для успіху проєкту?

ТЕМА 3.2
УПРАВЛІННЯ КОМУНІКАЦІЯМИ ПРОЄКТУ. УПРАВЛІННЯ ЧАСОМ ПРОЄКТУ

План

1. Канали комунікацій. Засоби зв’язку.
2. Документообіг у проєкті. Звітування.
3. Управління часом.
4. Методи оцінки тривалості завдань.
5. Календарне планування.
6. Критичний шлях.

Ключові слова: комунікація, зв’язок, документи, звіти, час, план, завдання, графік, критичний шлях, ефективність

Key words: communication, connection, documents, reports, time, plan, tasks, schedule, critical path, efficiency

1. Канали комунікацій. Засоби зв’язку

1.1. Поняття комунікацій у проєкті та їхнє значення

Комунікація в ІТ-проєкті – це процес обміну інформацією між членами команди, замовниками та іншими зацікавленими сторонами з метою забезпечення узгодженої роботи над продуктом. Йдеться не просто про передачу повідомлень, а про досягнення спільного розуміння.

Управління комунікаціями охоплює планування каналів зв’язку, визначення відповідальних, вибір інструментів та створення правил взаємодії. У ІТ-сфері, де зміни постійні, комунікація – це постійний процес підтримки синхронності між людьми.

1.2. Канали комунікації: їхні типи та природа

Комунікаційні канали можна поділити на різні типи залежно від формату, швидкості, синхронності та глибини взаємодії. Щоб краще зрозуміти їхню природу, варто розглянути їх у контексті реальних робочих ситуацій.

Формальні канали. Формальна комунікація включає офіційні зустрічі, документацію, звіти, листування електронною поштою. В ІТ-проєктах такі канали допомагають фіксувати важливі рішення, узгодження, протоколи. Вони створюють слід, який можна переглянути в будь-який момент. Це своєрідна «юридична» частина взаємодії, де кожне слово має точне значення.

Неформальні канали. До них належать повідомлення в месенджерах, короткі розмови, уточнення під час стендапів, обговорення в чатах. Ці канали розвантажують процес і роблять його швидким. Вони важливі для оперативності, але якщо їх використовують неправильно – інформація може загубитися.

Синхронні та асинхронні канали. Синхронна комунікація – це спілкування в реальному часі: зустрічі, дзвінки, відеоконференції. Асинхронна – це все, що не вимагає негайної відповіді: email, тікети, коментарі в системах управління завданнями.

Вдалий баланс між цими типами дозволяє команді працювати ефективно, без перенавантаження та затримок.

1.3. Засоби зв'язку в IT-проєктах

Сучасні IT-команди використовують широкий спектр комунікаційних інструментів. Вибір залежить від типу проєкту, культури компанії, складу команди та характеру завдань. Розгляньмо основні групи засобів зв'язку.

Інструменти для оперативної комунікації. Месенджери (Slack, Microsoft Teams, Discord, Telegram) дозволяють швидко реагувати на питання. Вони замінюють короткі особисті розмови. Проте важливо створити правила їх використання, наприклад: канали за темами, маркування важливих повідомлень, обмеження офтопу.

Інструменти для стратегічної та формальної комунікації. Електронна пошта та офіційні документи використовуються для узгоджень, надсилання протоколів, юридично важливих деталей або тривалих пояснень. Це більш повільний, але стабільний канал.

Системи управління завданнями. Jira, Trello, Asana, ClickUp – це не просто засоби для створення тікетів. Це канали передачі вимог, статусів, пріоритетів і зворотного зв'язку. Через них команда фіксує роботу.

Інструменти для зустрічей і колаборацій. Zoom, Google Meet, Miro, Figma. Вони дають можливість спільно працювати над матеріалами у реальному часі. Відеозустрічі допомагають передати емоційний контекст і уникнути непорозумінь.

Тип каналу	Приклади	Переваги	Ризики
Синхронні	Дзвінки, Zoom, стендапи	Швидкість, ясність	Перевантаження зустрічами
Асинхронні	Email, Jira, Trello	Прозорість, архівування	Затримки у відповідях

Формальні	Документи, звіти	Чіткість, фіксація рішень	Більше часу на створення
Неформальні	Чати, голосові нотатки	Гнучкість, швидкість	Загублена інформація

Таблиця 3.2.1. Таблиця-порівняння каналів комунікацій.

1.4. Вибір каналів у різних ситуаціях

Проєктний менеджер повинен вибирати канал залежно від складності питання, рівня терміновості, рівня ризику, потреби в доказовості, технічної глибини інформації

Наприклад, складне технічне завдання краще передати через документ + короткий дзвінок, а дрібне питання можна обговорити у Slack.

Життєвий приклад: якщо дизайнеру потрібно уточнити колір кнопки – чат ідеальний. Але якщо потрібно узгодити зміни в UI всього модуля – краще провести коротку зустріч із подальшим документом у Confluence.

Найпоширенішими проблемами в комунікації IT-команд є:

- інформація, що фіксується не там, де її очікують інші;
- рішення приймається усно, але не документується;
- надлишок зустрічей виснажує команду;
- відсутність єдиних правил використання інструментів;
- спроба «рятувати» все чатами;
- невиразність формулювань завдань.

Ці проблеми часто призводять до плутанини, дублювання роботи й пропущених деталей.

2. Документообіг у проєкті. Звітування

2.1. Документообіг як основа керованості проєкту

Документообіг можна визначити як організований процес створення, зберігання, оновлення, поширення та контролю документації, яка супроводжує проєкт на всіх його етапах.

Він охоплює як стратегічні документи (паспорт проєкту, план управління ризиками, контракти), так і оперативні (технічні специфікації, щоденні нотатки, звіти про помилки, журнали змін, протоколи зустрічей). Усе це формує єдиний інформаційний простір, до якого мають доступ члени команди й стейкхолдери.

Документи виконують кілька ключових ролей: забезпечують фіксацію домовленостей, синхронізують роботу команди, підтримують стандарти якості

та дозволяють обґрунтовувати рішення. У разі конфлікту або непорозуміння завжди є до чого повернутися – не на рівні емоцій, а на рівні фактів.

2.2. Основні групи проєктної документації

Хоча кожен проєкт має свій унікальний набір документів, можна виокремити кілька основних груп, які притаманні більшості IT-ініціатив.

1. Стратегічна документація. Це документи, які визначають рамки проєкту: що ми робимо, навіщо, хто залучений, які обмеження та очікування. До них належать паспорт проєкту, статут, опис цілей, переліки ризиків, логічні моделі.

2. Планувальна документація. Вона описує маршрути досягнення результатів: календарні плани, бюджет, WBS, перелік задач, дорожні карти релізів, план забезпечення якості, план комунікацій.

3. Технічна документація. Це інструкції, технічні специфікації, архітектурні діаграми, вимоги, API-описання, документація до продукту.

4. Оперативна документація. Сюди входять щоденні нотатки, журнали помилок, стендап-звіти, протоколи зустрічей, рішення нарадами, журнали змін.

5. Звітна документація. Звіти про виконання задач, фінансові звіти, звіти KPI, підсумкові аналізи спринтів, фінальні звіти проєкту.

2.3. Вимоги до якісної проєктної документації

Документи повинні не просто існувати, а **виконувати функцію**, тому головними критеріями їхньої якості є ясність, актуальність, доступність та контрольованість.

Документація втрачає цінність, якщо вона застаріла, її складно знайти, вона не відповідає реальним процесам, у команді немає єдиного формату записів, а також коли частина інформації існує лише у вигляді усних домовленостей.

Усе це створює непорозуміння, конфлікти й затримки.

Тому правило якісної документації звучить так: «достатньо, але не надмірно». IT-командам не потрібні сотні сторінок тексту – потрібно чітко, коротко й по суті. Концепція lean-documentation (мінімальної документації) особливо добре працює в Agile-середовищі.

2.4. Організація документообігу в сучасних IT-проєктах

Сучасні команди використовують цифрові системи керування документацією. Це можуть бути Confluence, Notion, Google Workspace, Jira, ClickUp та інші платформи.

Суть цих інструментів не лише у зберіганні документів, а в тому, що вони дозволяють вести історію змін, працювати над документами колективно, налаштовувати структуру й шаблони, обмежувати доступи, створювати зв’язки між сторінками та артефактами, зберігати документацію поруч із задачами.

Наприклад, у Scrum команда може мати єдиний Confluence-простір, де зберігаються Product Backlog, технічні специфікації, результати ретроспектив і

протоколи демо. Це дозволяє новому співробітнику швидко адаптуватися, а менеджеру – контролювати актуальність інформації.

2.5. Звітування як форма комунікації та контролю

Звітування – це процес систематичного фіксування та передачі інформації про стан проєкту стейкхолдерам. Його головне завдання – забезпечити прозорість і можливість ухвалення рішень.

Звіт – це не бюрократія, а засіб підтримувати довіру між командою й замовником.

В IT-проєктах звіти бувають:

- регулярними (щотижневий статус-репорт, спринт-репорт),
- оперативними (повідомлення про ризики, блокери),
- фінальними (підсумковий звіт про виконання проєкту).

Якісний звіт повинен містити: статус задач, прогрес, план на наступний період, проблеми, ризики та пропозиції.

У багатьох командах зручним є формат «зелений – жовтий – червоний», де кожен блок позначає рівень стабільності проєкту. Наприклад, зелений – усе за планом, жовтий – є ризики, червоний – проєкт потребує втручання.

Блок звіту	Зміст
Поточний статус	Прогрес виконання задач у відсотках
Досягнення	Що було завершено цього тижня
План на наступний тиждень	Ключові задачі
Ризики	Загрози та ймовірність їх реалізації
Потреби	Що потрібно від стейкхолдерів

Таблиця 3.2.1. Приклад формату простого щотижневого звіту.

Такий формат забезпечує структурованість, але залишається компактним.

2.6. Значення документообігу та звітування

Документація й звіти – це фундаментальні інструменти управління проєктом, вони:

- створюють прозорість;
- зменшують ризики;
- забезпечують сталість знань;
- допомагають новим членам команди швидко включатися;

- підтримують довіру з боку замовника;
- дають змогу відстежувати зміни та обґрунтовувати рішення.

У великих IT-проєктах саме документація часто є тим фактором, який визначає, чи можна масштабувати команду, чи команда «прив'язана» до кількох ключових людей.

3. Управління часом

3.1. Управління часом як процес

Під управлінням часом розуміють сукупність процесів, спрямованих на планування, оцінку, контроль і коригування строків виконання проєкту. Це не лише складання графіка, а постійна робота з аналізу залежностей, визначення критичних точок, розуміння можливих затримок і пошуку шляхів оптимізації.

Для IT-проєктів час часто стає ресурсом, який неможливо повернути. Технології змінюються, очікування користувачів зростають, конкуренти виходять на ринок швидше. Якщо команда не встигає адаптуватися, вона втрачає не лише темп, а й перевагу.

3.2. Чому управління часом складне в IT-сфері

Головною складністю є непередбачуваність задач. Наприклад, виправлення одного бага може зайняти годину або два дні. Розробка нової функції може «поїсти» половину спринту, хоча здавалося простою. Тому час у IT потрібно розглядати не як точну величину, а як керований ризик.

Управління часом включає:

- розбиття проєкту на дрібні складові;
- оцінку тривалості кожної частини;
- визначення залежностей;
- виявлення критичного шляху;
- складання календарного плану;
- моніторинг прогресу та коригування.

Менеджер працює не лише з датами, а з очікуваннями, конфліктами пріоритетів, обмеженнями ресурсів і невизначеністю вимог.

3.3. Формування основи для управління часом: WBS і структурування робіт

Перед тим як оцінювати строки, команда повинна чітко розуміти перелік усіх завдань. Це робиться через ієрархічну структуру робіт (WBS). WBS допомагає побачити обсяг роботи, розділити складні частини на малі, зрозуміти взаємозв'язки й підготуватися до реалістичної оцінки часу.

Управління часом починається саме з WBS, бо неможливо оцінити те, що не визначено.

4. Методи оцінки тривалості завдань

Оцінка тривалості – фундамент управління часом. Вона дозволяє передбачити терміни, скласти реалістичний план роботи та уникнути перевантаження команди. Існують різні методи оцінки, кожен з яких має свої переваги:

1. Експертна оцінка. Один із найпоширеніших підходів. Досвідчені розробники, тестувальники або аналітики оцінюють завдання на основі власного досвіду. Метод швидкий, але залежить від людського фактора.

Коли підходить: у невеликих командах, у повторюваних типах задач.

2. Метод трьох точок (PERT). Метод, який дозволяє врахувати невизначеність. Він оцінює три вартості:

- оптимістичну (O)
- песимістичну (P)
- найімовірнішу (M)

Формула: (O + 4M + P) / 6

Такий підхід дозволяє побудувати більш реалістичний прогноз, враховуючи ризики.

3. **Аналогічне оцінювання**. Оцінка на основі подібних задач у минулих проєктах. Якщо команда вже робила схожу функцію або модуль, можна порівняти їх за складністю.

Перевага: швидкість

Недолік: неточність у нових, інноваційних проєктах

4. **Bottom-up оцінювання.** Оцінюється кожна дрібна підзадача, після чого загальна тривалість складається з їх суми. Найточніший метод, але потребує великої деталізації.

Зазвичай використовується разом із WBS.

5. **Planning Poker (Agile).** Метод групової оцінки, у якому команда встановлює складність задач у вигляді «story points». Це не час, а відносна складність. Потім story points переводяться в реальні строки. Метод працює добре, бо сприяє командному вирівнюванню очікувань.

Після оцінювання часу настає етап планування – складання календаря, визначення залежностей, встановлення дедлайнів. Але реальне управління часом триває протягом усього проєкту.

Менеджер постійно:

- аналізує відхилення від графіка;
- переглядає пріоритети;
- узгоджує зміни;
- комунікує ризики;

- коригує план.

Управління часом в IT – це динамічний процес, який вимагає гнучкості. Навіть добре складений план змінюється під впливом нових даних, технічних перешкод або коригування вимог.

Люди мають тенденцію недооцінювати складність задач – явище, яке називають помилкою планування. Особливо це виражено у сфері програмування, де «дрібна» правка може перетворитися на великий технічний борг.

Тому важливо не робити оцінки поспіхом, залучати команду, аналізувати минулі спринти, будувати буфери часу.

Управління часом – це баланс між оптимізмом та реальністю.

5. Календарне планування

5.1. Поняття календарного планування

Календарне планування – це процес визначення послідовності завдань, їх тривалості, взаємозв'язків та реальних дат початку і завершення, який дозволяє побудувати повну часову модель проєкту. Фактично це відображення логіки проєкту в часовій шкалі.

Календарний план допомагає менеджеру відповісти на ключові питання

- Коли проєкт буде завершено?
- Які роботи мають бути виконані першими?
- Які завдання можуть виконуватися паралельно?
- Де знаходяться ризики затримок?

Інколи календарний план плутають із простою діаграмою Ганта, але насправді це ширше поняття. Діаграма – лише спосіб візуалізації. Календарне планування включає логіку залежностей, розподіл ресурсів, оцінки тривалостей, визначення критичного шляху і навіть врахування зовнішніх факторів: відпустки, свята, графіки підрядників, доступність інженерів.

5.2. Основні елементи календарного плану

Календарний план будується на основі WBS, оцінок тривалостей, ресурсів і технологічних залежностей. До його ключових елементів належать:

- логічні залежності між роботами,
- тривалість кожного завдання,
- доступність виконавців,
- календар ресурсу,
- критичні точки та буфери часу.

У практиці проєктного менеджменту побудова календарного плану нагадує складання пазла, де кожен елемент має своє місце. Менеджер поступово

розкладає завдання у часі, перевіряючи, чи не виникають конфлікти ресурсів, перекриття або затори.

Уявімо приклад із життя: ремонт квартири. Неможливо замінювати проводку, коли вже пофарбовані стіни; плитку не кладуть до того, як зроблена гідроізоляція; меблі не приїдуть раніше, ніж завершиться підлога. Усе має свою логіку. У проєкті з розробки ПЗ – те саме: не можна тестувати функцію, доки вона не написана; не можна писати код, коли вимоги не узгоджені; не варто розробляти дизайн без архітектури.

5.3. Інструменти календарного планування

Сучасний менеджер ІТ-проєкту має у своєму розпорядженні велику кількість інструментів для планування. Найпоширеніші:

- **MS Project** – класичний інструмент з діаграмами Ганта, мережевими моделями й критичним шляхом.
- **Jira Advanced Roadmaps** – підходить для Agile-команд, дозволяє планувати спринти, релізи та епіки.
- **Notion / Asana / Monday.com** – універсальні платформи з календарями та таймлайнами.
- **GanttPRO, Wrike** – онлайн-інструменти зі зручними діаграмами.

Планування в ІТ часто поєднує традиційні та гнучкі інструменти: спринти Scrum можуть доповнюватися високорівневими діаграмами Ганта для довгострокового бачення.

6. Критичний шлях

6.1. Суть критичного шляху

Критичний шлях (Critical Path Method, CPM) – це послідовність робіт, що не мають резерву часу. Це найдовший шлях від початку до завершення проєкту, що визначає мінімально можливу тривалість усього проєкту. Завдання, що лежать на критичному шляху, називаються критичними, бо їхня затримка автоматично затримує весь проєкт.

Як тільки одне з таких завдань затримується, весь проєкт "пливе". Саме тому керівники проєктів приділяють критичним завданням найбільшу увагу.

6.2. Види залежностей у проєкті

Щоб визначити критичний шлях, менеджер формує мережеву модель на основі залежностей між роботами. Найпоширеніші типи:

- **Finish-to-Start (FS)** – наступне завдання починається лише після завершення попереднього (найпоширеніша залежність).
- **Start-to-Start (SS)** – завдання можуть починатися паралельно.

- **Finish-to-Finish (FF)** – завдання завершуються приблизно одночасно.
- **Start-to-Finish (SF)** – рідкісний тип, у специфічних сценаріях.

Ці залежності дозволяють побудувати мережеву діаграму, яка й використовується для пошуку критичного шляху.

6.3. Чому критичний шлях важливий в IT

IT-проєкти мають багато паралельних потоків робіт. Архітектура, дизайн, бекенд, фронтенд, тестування, DevOps – все це може йти одночасно, але деякі речі залежать одна від одної. Саме критичний шлях дозволяє побачити де ризики затримок, які задачі потрібно контролювати щодня, які ресурси потрібні саме зараз, як можна скоротити час проєкту (crashing, fast-tracking).

Питання для самоконтролю

1. Що таке комунікації у проєкті?
2. Чому ефективна комунікація є ключовою для успіху проєкту?
3. Які існують канали комунікацій у проєкті?
4. Які засоби зв’язку найчастіше використовуються у проєктних командах?
5. Що таке документообіг у проєкті?
6. Які документи найчастіше створюються у проєкті?
7. Навіщо потрібне регулярне звітування?
8. Що таке управління часом у проєкті?
9. Які етапи включає процес управління часом?
10. Які дані необхідні для оцінки тривалості завдань?
11. Які методи оцінювання тривалості є найбільш поширеними?
12. Що таке календарне планування?
13. У чому полягає різниця між графіком робіт і календарним планом?
14. Як діаграма Ганта допомагає у плануванні часу?
15. Що означає поняття «критичний шлях»?
16. Чому визначення критичного шляху важливе для керівника проєкту?
17. Як затримка одного завдання може вплинути на весь проєкт?
18. Чому комунікації та управління часом тісно пов’язані у роботі менеджера проєкту?

ТЕМА 3.3
УПРАВЛІННЯ РИЗИКАМИ ПРОЄКТУ

План

1. Поняття ризику.
2. Класифікація ризиків.
3. Методи ідентифікації ризиків.
4. Аналіз і оцінка ризиків.
5. План реагування на ризики.
6. Контроль ризиків. Зниження впливу ризиків.

Ключові слова: ризик, ідентифікація, оцінка, аналіз, управління, план, контроль, вплив, реагування, невизначеність

Key words: risk, identification, assessment, analysis, management, plan, control, impact, response, uncertainty

1. Поняття ризику

Ризик у проєкті – це подія, яка може статися в майбутньому та негативно або позитивно вплинути на проєкт. У більшості випадків у контексті управління ризиками говорять про негативні наслідки, але позитивні ризики – можливості – також є частиною системи планування.

Ризик завжди пов'язаний із невизначеністю. Якщо ми точно знаємо, що подія станеться, це вже не ризик, а проблема. Проблеми треба вирішувати, а ризики – передбачати.

В ІТ проєктах ризики виникають через специфіку галузі: складні технічні рішення, залежність від людей і процесів, швидкі зміни вимог, інтеграцію з іншими системами, наявність зовнішніх постачальників, потребу у високій якості. Тому менеджер ІТ-проєкту повинен мати «радар ризиків» – здатність прогнозувати можливі відхилення ще до того, як вони заподіють шкоду.

Ключові властивості ризику:

- **Ймовірність** – шанс того, що подія трапиться.
- **Вплив (impact)** – наскільки сильно подія зачепить бюджет, строки, команду, якість.
- **Індикатори ризику** – ознаки, які свідчать, що ризик може почати реалізовуватися.

- Умовно ризик можна уявити як насуваючу грозу: можливо, пройде повз; можливо, буде легкий дощ; а можливо, зірве дах. Завдання менеджера – не чекати, що буде «якось», а підготуватися до кожного варіанта заздалегідь.
- Ризики в IT можуть впливати на:
- **Строки** (затримки релізів, перевищення тривалості задач).
- **Бюджет** (додаткові витрати на інструменти, людей, навчання).
- **Якість** (дефекти, технічний борг, невідповідність стандартам).
- **Команду** (вигорання, конфлікти, нестача експертизи).
- **Клієнта** (незадоволеність, зміна пріоритетів).

Це дозволяє зрозуміти, що ризик – не абстракція, а реальна подія, яка може проявитися у різних площинах проєкту.

Ризики супроводжують проєкт на всіх етапах, але саме на стадії ініціації та планування їх можна передбачити найбільш ефективно.

2. Класифікація ризиків

Класифікація дозволяє менеджеру структурувати ризики, що значно полегшує роботу з ними. В IT-проєктах ризики зазвичай групують за джерелом виникнення, за природою впливу та за характером наслідків.

Розглянемо найбільш корисні класифікації для практичного застосування:

- **Класифікація за джерелом виникнення**

Це найпоширеніший підхід, який дозволяє зрозуміти, звідки може прийти загроза.

Категорія	Опис	Приклад
Технічні ризики	Пов’язані з технологіями, архітектурою, інфраструктурою	Вибір нестабільної бібліотеки; проблеми зі швидкодією
Організаційні ризики	Невизначеність у процесах, ролях, управлінні	Зміна пріоритетів у керівництва, відсутність власника продукту
Людські ризики	Команда, компетенції, мотивація, комунікація	Хвороба ключового фахівця, конфлікт у команді
Зовнішні ризики	Фактори, на які команда не може впливати	Збої постачальника, зміни в законодавстві

Ризики клієнта	Невизначеність щодо вимог чи рішень замовника	Запізніле погодження макетів, зміна візії продукту

Ця класифікація корисна на початку проєкту, коли менеджер створює реєстр ризиків.

- **Класифікація за впливом на проєкт**

Ризики можуть впливати по-різному, тому важливо розуміти, що саме вони змінюють.

  - Ризики строків – затримки, залежності, перевантаження команди.
  - Ризики вартості – бюджетні перевищення, додаткові закупівлі.
  - Ризики якості – дефекти, обмеженість тестування, технічний борг.
  - Ризики змісту (scope risks) – розширення обсягу робіт, зміни вимог.
  - Ризики комунікацій – втрата інформації, непорозуміння.

Цей підхід дозволяє визначити, які саме цілі проєкту під загрозою.

- **Класифікація за ймовірністю і масштабом впливу**

Цю класифікацію зручно використовувати разом із матрицею ризиків (таблиця 3.3.2.).

	Низький вплив	**Середній вплив**	**Високий вплив**
Низька ймовірність	Низький ризик	Низький/середній	Середній
Середня ймовірність	Низький/середній	Середній	Високий
Висока ймовірність	Середній	Високий	Критичний

Таблиця 3.3.2. Матриця ризиків.

Найнебезпечніші – події з високою ймовірністю та високим впливом. Саме на них менеджер витрачає найбільше уваги.

- **Позитивні та негативні ризики**

Управління проєктами розглядає ризик не тільки як загрозу, а й як потенціал.

- Негативний ризик (threat) – небажана подія, яку потрібно уникнути або зменшити.
- Позитивний ризик (opportunity) – подія, яка може дати додаткову цінність: скорочення часу, економію коштів, покращення продукту.

Приклад позитивного ризику: команда випадково знаходить нову бібліотеку, яка дозволяє пришвидшити розробку на 20%. Це можливість, яку варто використати.

- **Класифікація за стадіями життєвого циклу**

Ризики різні на різних етапах:

- На ініціації найбільше стратегічних ризиків.
- На плануванні – організаційних та технічних.
- На виконанні – людських та якісних.
- На завершенні – ризики приймання й релізу.

Це дозволяє менеджеру фокусуватися на найважливішому саме зараз.

3. Методи ідентифікації ризиків

Ідентифікація ризиків – це процес виявлення всіх потенційних подій, які можуть негативно вплинути на проєкт. Головна думка: ризик, який виявлений, менше шкодить, ніж ризик, який не усвідомлений. Завдання менеджера – зібрати та структурувати якомога повніший перелік ризиків, не пропускаючи навіть малоймовірні події, які можуть мати значні наслідки.

Одним із найефективніших способів є **аналітичний огляд досвіду попередніх проєктів**. Таймлайни, звіти, помилки, архівні документи – усе це допомагає побачити повторювані проблеми. У компаніях, де ведеться база знань, менеджери мають змогу звертатися до історії: наприклад, якщо попередній проєкт затримувався через інтеграцію сторонніх API, це ризик і для нового проєкту.

Метод мозкового штурму – це структурована робота з командою, яка дозволяє зібрати максимально повний перелік думок. Тут важливо зібрати фахівців різних ролей: аналітиків, розробників, тестувальників, DevOps-інженерів. Кожен бачить проєкт під іншим кутом, тому ризики будуть різні.

Контрольні списки (check-lists) – це зручний інструмент, особливо в організаціях із усталеними процесами. Список включає типові ризики: кадрові, технічні, фінансові, документаційні, ризики комунікацій і залежностей.

Діаграма Ішікави (риб’яча кістка) – інструмент, що дозволяє візуально розкласти проблему на складові: люди, процеси, інструменти, середовище. Це допомагає знайти приховані джерела ризиків.

4. Аналіз і оцінка ризиків

Після ідентифікації ризиків необхідно виконати їхню оцінку та визначити ступінь небезпеки. Аналіз ризику допомагає зрозуміти, наскільки той чи інший ризик вплине на строки, бюджет, якість або команду. Важливо визначити, які ризики потрібно контролювати першочергово, а які – можна просто відстежувати.

1. Якісний аналіз

Якісний аналіз не передбачає точних числових значень. Менеджер та команда оцінюють ризики за такими параметрами:

- ймовірність настання (низька, середня, висока),
- ступінь впливу (незначний, середній, критичний),
- терміновість реагування,
- можливість попередження.

На основі цього формується матриця ризиків, яка дозволяє швидко визначити зони найбільшої небезпеки.

	Низький вплив	**Середній вплив**	**Високий вплив**
Низька ймовірність	Прийняти	Відстежувати	Контролювати
Середня ймовірність	Відстежувати	Контролювати	Зменшувати
Висока ймовірність	Контролювати	Мінімізувати	Негайно реагувати

Таблиця 3.3.4. Матриця якісної оцінки ризиків (приклад).

Це простий, але дуже ефективний інструмент.

2. Кількісний аналіз

Кількісний аналіз прагне надати ризикам точні числові показники. Він використовується в великих або коштовних проєктах. Оцінка включає:

- розрахунок можливих втрат у грошовому еквіваленті;
- моделювання сценаріїв (best-case, worst-case);
- метод Монте-Карло (статистичні симуляції);
- оцінку впливу на критичний шлях.

Наприклад, якщо існує ризик того, що ключовий спеціаліст може захворіти, у кількісному аналізі цей ризик оцінюють у втрачених людино-днях, грошових витратах та впливі на дату релізу.

3. Ранжування ризиків

Після оцінки формується список ризиків, відсортований за пріоритетом впливу на проєкт. Це дозволяє розподілити увагу та ресурси на ті події, які є найбільш критичними.

5. План реагування на ризики

5.1. План реагування на ризики: сутність та значення

Після того, як ризики були проаналізовані й оцінені, команда переходить до створення плану реагування на ризики. Це системний документ або набір стратегій, який визначає, яким чином проєкт реагуватиме на кожен ризик. Він не лише описує дії, а й закріплює відповідальних осіб, необхідні ресурси, строки виконання та критерії успішності. У практиці управління ІТ-проєктами план реагування є продовженням ризик-реєстру та невід'ємною частиною проєктної документації.

Головна мета плану – зменшення ймовірності або наслідків ризику. Або і того, й іншого одночасно. Це не означає, що проєкт позбавляється ризиків повністю. Це означає, що команда підготовлена, має сценарій дій і здатна зменшити втрати.

5.2. Основні стратегії реагування на ризики

У практиці ІТ-проєктів існує кілька базових стратегій, які застосовуються залежно від типу ризику та його критичності. Найважливіші з них:

- **Уникнення ризику.** Команда змінює план або підхід таким чином, щоб ризик взагалі не виник. Наприклад, якщо вибір конкретної технології може спричинити затримки, команда обирає іншу, більш стабільну.
- **Зменшення ризику (мітігація).** Проєкт не усуває ризик повністю, але знижує ймовірність його появи або силу впливу. Наприклад, додаткове тестування, навчання розробників, проміжні прототипи.
- **Передача ризику.** Відповідальність перекладається на іншу сторону – постачальника, підрядника, хмарний сервіс тощо. Типовий приклад: передача DevOps-обслуговування партнерській компанії.
- **Прийняття ризику.** Команда визнає ризик і нічого не робить з ним заздалегідь. Така стратегія застосовується для малоймовірних або незначних ризиків. Але при цьому створюється резерв плану або резерв бюджету.

6. Контроль ризиків. Зниження впливу ризиків.

6.1. Контроль ризиків

Контроль ризиків – це безперервний процес, який триває від першого дня проєкту до моменту його завершення. Він включає відстеження виявлених ризиків, моніторинг нових ризиків, контроль ефективності стратегій реагування та оновлення ризик-реєстру.

У реальних IT-командах контроль ризиків здійснюється під час кожної зустрічі: на плануваннях, стендапах, технічних обговореннях, оцінках пріоритетів. Це не окремий процес «на папері», а частина щоденної роботи. Наприклад, якщо під час спринту з'являються затримки через непередбачені технічні труднощі, Scrum-команда оперативно переглядає пріоритети та оновлює ризикові записи.

До основних інструментів контролю належать:

- регулярні перегляди ризик-реєстру;
- статус-зустрічі з обговоренням ризиків;
- метрики (velocity, час виконання, кількість дефектів);
- burn-down та burn-up діаграми;
- аналіз варіацій у календарному плані;
- постійний моніторинг критичних залежностей.

Це формує культуру, у якій ризики не ігноруються, а аналізуються.

6.2. Зниження впливу ризиків: превентивні заходи

Зниження впливу ризиків означає створення умов, за яких навіть при настанні ризику проєкт залишається життєздатним. У IT-практиці превентивні заходи можуть включати удосконалення процесів, підвищення компетентності команди, використання автоматизації, дублювання важливих систем або створення технічних резервів.

Наприклад, для зниження технічних ризиків компанії часто проводять архітектурні рев'ю, додають автоматичні тести, використовують CI/CD-системи, створюють staging-середовища. Для зниження ризиків комунікації – впроваджують регулярні демо, прозорі канали зв'язку, структурують документацію.

Превентивні заходи також можуть враховувати людський фактор. Якщо проєкт залежить від одного сильного розробника, це великий ризик. Тому менеджер може запровадити практику парного програмування або документування складних частин системи.

6.3. Контроль ефективності реагування на ризики

Контроль ризиків – це не тільки відстеження, а й визначення, наскільки обрані стратегії працюють. Якщо ризик продовжує проявлятися або його наслідки збільшуються, це сигнал, що план реагування потребує корекції. Це своєрідний «цикл зворотного зв'язку», характерний для Agile-методологій.

Менеджери часто використовують матрицю «ризик–ефективність реакції» (таблиця 3.3.6.), яка дозволяє зрозуміти, чи правильні дії були обрані.

Ризик	План реакції	Результат	Оцінка ефективності
Затримка бекенд-розробки	Залучення ще одного розробника	Термін виконання скорочено	Висока
Нерозуміння вимог клієнтом	Часті зустрічі	Вимоги стали чіткішими	Середня
Перевантаження тестувальників	Автоматизація	Зниження ручної роботи	Висока
Технічний борг	Код-рев'ю	Проблеми залишилися	Низька (потрібні зміни)

Таблиця 3.3.6. Приклад матриці «ризик–ефективність реакції».

Питання для самоконтролю

1. Що таке ризик у проєкті?
2. Чому ризики є невід'ємною частиною будь-якого проєкту?
3. Які фактори можуть спричинити появу ризиків?
4. Як класифікують ризики у проєктній діяльності?
5. Чим відрізняються внутрішні ризики від зовнішніх?
6. Що таке методи ідентифікації ризиків?
7. Які методи ідентифікації ризиків використовуються найчастіше?
8. Чому важливо залучати команду до процесу виявлення ризиків?
9. Що означає аналіз ризиків?
10. У чому полягає якісна оцінка ризиків?
11. Що таке кількісна оцінка ризиків?
12. Навіщо створюють матрицю ризиків?
13. Що таке план реагування на ризики?
14. Які стратегії реагування використовуються для негативних ризиків?
15. Які стратегії реагування існують для позитивних ризиків (можливостей)?
16. Чому важливо здійснювати постійний контроль ризиків?
17. Як можна знизити вплив ризику на проєкт?
18. Чому управління ризиками повинно проводитися протягом усього життєвого циклу проєкту?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Березін О. В., Безпарточний М. Г. Управління проектами : навч. посіб. Суми : Університетська книга, 2023. 272 с.
2. Бізнес-планування та управління проектами : навч. посіб. / П. Г. Ільчук, Р. В. Фещкр, А. І. Якимів та ін. ; за ред. П. Г. Ільчука. Львів : Новий Світ-2000, 2023. 240 с.
3. Блага Н. В. Управління проєктами : навч. посіб. Львів : ЛДУВС, 2021.152 с.
4. Дворжак В. В., Томка Ю. Я. Управління ІТ-проектами. Частина 1: Бізнес-аналіз та ініціація проекту. Чернівці : Технодрук, 2022. 521 с.
5. Добровська Л. М., Аверьянова О. В. Управління ІТ-проектами в Microsoft Project. Комп'ютерний практикум : навчальний посібник. Київ : КПІ ім. Ігоря Сікорського, 2020. 152 с. URL: https://ela.kpi.ua/handle/123456789/33622
6. Єгорова О. В. Методичні рекомендації до виконання лабораторних робіт з дисципліни «Управління ІТ-проектами» для здобувачів освітнього ступеня «бакалавр» зі спеціальностей 122 Комп'ютерні науки та 126 Інформаційні системи та технології. Черкаси : ЧДТУ, 2019. 63 с.
7. Катренко А. В. Управління ІТ-проектами. Книга 1. Стандарти, моделі та методи управління проектами : підручник. Львів : Новий Світ – 2000, 2025. 550 с.
8. Когут І. В., Ільчук П. Г., Якимів А. І. Управління командою проекту : навч. посіб. Львів : Новий Світ-2000, 2023. 154 с.
9. Конінг П. Інструментарій agile-лідера : учимося успішно розвиватися за допомогою самокерованих команд / пер. з англ. В. Луненко. Харків : Видавн. дім Фабула, 2023. 224 с.
10. Корчак Н. М., Обушна Н. І. Управління проєктами в публічній сфері : навч. посіб. Київ : Каравела, 2022. 272 с.
11. Крижановський Є. М., Ящолт А. Р., Жуков С. О. Моделювання бізнес-процесів та управління ІТ-проєктами : навчальний посібник. 2-ге вид., змін. та доповн. Вінниця : ВНТУ, 2022. 129 с. URL: https://pdf.lib.vntu.edu.ua/books/2023/Kryzhanov_2022_129.pdf
12. Кузьміних В. О., Тараненко Р. А. Основи управління ІТ проектами : навч. посіб. для студ. спеціальності 122 «Комп'ютерні науки». Київ : КПІ ім. Ігоря Сікорського, 2019. 75 с.
13. Микитюк П. П., Брич В. Я., Микитюк Ю. І., Труш І. М. Управління проектами : навч. посіб. Тернопіль : ЗУНУ, 2021. 416 с.
14. Ноздріна Л. В., Ящук В. І., Полотай О. І. Управління проектами : підручник / за заг. ред. Л.В. Ноздріної. Київ: Центр учбової літератури, 2010. 432 с.
15. Основи управління ІТ проектами [Електронний ресурс]: навч. посіб. / уклад.: В. О. Кузьміних, Р. А. Тараненко. Київ : КПІ ім. Ігоря Сікорського, 2019. 75 с. URL:

https://ela.kpi.ua/server/api/core/bitstreams/7c313e5c-5477-4be2-9806-d32e9eace0c3/content
16. Петренко Н. О., Кустріч Л. О., Гоменюк М. О. Управління проектами : навч. посіб. Київ : ЦУЛ, 2021. 244 с.
17. Словник термінів з управління проєктами PMI. Версія 3.3. Newtown : Project Management Institute, 2022. 25 с. URL: https://pmiukraine.org/lexicon
18. Строкань О. В., Мірошниченко М. Ю. Управління ІТ-проєктами : лабораторний практикум. Мелітополь: Видавничо-поліграфічний центр «Люкс», 2020. 135 с. URL: http://elar.tsatu.edu.ua/bitstream/123456789/14494/1/6.pdf
19. Управління ІТ-проєктами: Загальні питання теорії управління ІТ-проєктами : навчальний посібник / уклад. Л. М. Добровська, О. С. Коваленко, О. А. Аверьянова. Київ : КПІ ім. Ігоря Сікорського, 2022 284 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/1feb7c50-e0ef-4967-9611-997f2bb6d215/content
20. Хіґні Дж. Основи управління проєктами. Харків : Фабула, 2020. 272 с.
21. A Guide to the Project Management Body of Knowledge (PMBOK® Guide). 6th Edition. Project Management Institute, 2017. 756 p.

Навчальне видання

Управління ІТ-проєктами

Методичні рекомендації до лекційних занять
для змішаного навчання студентів
освітнього ступеня «Бакалавр»
за спеціальності F3 «Комп'ютерні науки»

Укладачі:

Ємельянов Святослав Ігорович
Тищенко Світлана Іванівна
Пархоменко Олександр Юрійович
Жебко Олександр Олегович
Богатєнкова Олександра Євгенівна

Формат 60х84 1/16. Ум. друк. арк. 9,00.
Наклад 50 прим. Зам. № _____

Надруковано у видавничому відділі
Миколаївського національного аграрного університету
54020, м. Миколаїв, вул. Георгія Гонгадзе, 9

Свідоцтво суб'єкта видавничої справи ДК № 4490
від 20.02.2013 р.