

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КІЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Моделі та засоби управління ІТ- проєктами

Навчальний посібник

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітньою програмою «Інженерія програмного забезпечення інтелектуальних
кібер-фізичних систем в енергетиці»
спеціальності 121 Інженерія програмного забезпечення

Укладачі: В. О. Кузьміних, О. В. Коваль, Р. А. Тараненко

Електронне мережне навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2023

Рецензент

Отроч Сергій Іванович, проф., д.т.н., кафедри цифрових технологій в енергетиці, ННІ АТЕ, «КПІ ім. Ігоря Сікорського»

Відповідальний
редактор

*Гавrilko Євген Володимирович, проф., д.т.н., заступник
директора з навчально-виховної роботи ННІ АТЕ, «КПІ ім.
Ігоря Сікорського»*

*Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 8 від 02.06.2023 р.)*

*за поданням Вченої ради ННІ АТЕ
(протокол № 13 від 25.05.2023 р.)*

Навчальний посібник «Моделі та засоби управління ІТ-проектами» створено для фахівців сфери інформаційних технологій та систем і призначено для оволодіння основами управління проектами. Матеріал викладено у обсязі необхідному в підготовці здобувачів ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». Посібник призначено для забезпечення проведення лекційних занять. Подано загальний та теоретичний матеріал, тематичний глосарій, розкрито зміст міжнародного стандарту сертифікації системи управління проектами ISO 21500. В навчальному посібнику розглянуто основи керування версіями програмного забезпечення – систем контролю версій (СКВ) для управління ІТ-проектами на прикладі широко вживаної системи GitHub. Навчальний посібник може бути корисним всім фахівцям з ІТ.

Реєстр. № НП 22/23-758. Обсяг 15,79 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Перемоги, 37, м. Київ, 03056
<https://kpi.ua>

Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів
і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

© КПІ ім. Ігоря Сікорського, 2023

ЗМІСТ

ЗМІСТ	3
ВСТУП	8
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ТА АКРОНІМІВ	10
РОЗДІЛ 1. ОСНОВНІ ПОНЯТТЯ. ОСОБЛИВОСТІ ІТ-ПРОЄКТІВ	13
1.1. Поняття «проект»	13
1.2. Місце управління проектами у процесі розробки програмних засобів ..	14
1.3. Інвестиційний проект.....	15
1.4. Головні елементи проекту в ІТ	15
1.5. Процеси проєктування та поточна операційна діяльність	17
1.6. Класифікація типів проєктів	18
1.7. Компоненти проєкту	20
1.8. Блоки робіт Project Management	20
Питання до розділу 1	21
РОЗДІЛ 2. ОСНОВНІ ПОНЯТТЯ. ЖИТТЕВИЙ ЦИКЛ. МОДЕЛІ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ	22
2.1. Життєвий цикл проєкту	22
2.2. Стадії життєвого циклу проєкту	22
2.3. Основні фази життєвого циклу проєкту	23
Структури управління проектами.....	23
Питання до розділу 2	28
РОЗДІЛ 3. ПЛАNUВАННЯ ПРОЄКТУ. ПАСПОРТ ПРОЄКТУ	29
3.1. Ініціація проєкту.....	29
3.2. Спонсор проєкту.....	29
3.3. Затвердження проєкту	29
3.4. Формування паспорт проєкту	30
Питання до розділу 3	33

РОЗДІЛ 4. ПЛANI ПРОЄКТУ. СТРУКТУРНА ДЕКОМПОЗИЦІЯ РОБІТ	
.....	34
4.1. Планування проєкту	34
4.2. Основні роботи менеджера проєкту по плануванню	35
4.3. Структурна декомпозиції робіт (СДР - WBS)	36
4.4. Робочий пакет	39
Питання до розділу 4	42
РОЗДІЛ 5. ФОРМУВАННЯ ЗМІСТУ ПРОЄКТУ	43
5.1. Зміст проєкту	43
5.2. Формування вимог	44
5.3. Методи збору інформації щодо очікувань та вимог замовника	44
5.4. Балансування вимог	46
5.5. Концепція проєкту (scope).....	47
Питання до розділу 5	49
РОЗДІЛ 6. ВИКОРИСТАННЯ ДІАГРАМИ ГАНТА ДЛЯ	
ПЛАНУВАННЯ РОБІТ ПРОЄКТУ	50
6.1. Призначення діаграми Ганта.....	50
6.2. Склад діаграми Ганта.....	51
6.3. Побудова діаграми Ганта	52
6.4. Приклад побудови діаграми Ганту	52
Питання до розділу 6	59
РОЗДІЛ 7. ЗВ'ЯЗКИ ТА КОМУНІКАЦІЇ ПРОЄКТУ	60
7.1. Управління комунікаціями проєкту (Project Communication Management)	
.....	60
7.2. Основні питання в управлінні комунікаціями.....	60
7.3. Планування системи комунікацій.....	61
7.4. Звітність про хід виконання проєкту.....	62
7.5. Інформаційна система управління проєктом.....	63
Питання до розділу 7	67

РОЗДІЛ 8. УПРАВЛІННЯ РИЗИКАМИ У ПРЄКТАХ	68
8.1. Поняття ризику та управління ризиками	68
8.2. Аналіз проектних ризиків	72
8.3. Методи зниження ризиків	72
8.4. Автоматизовані методи оцінки ризиків	74
Питання до розділу 8	78
РОЗДІЛ 9. УПРАВЛІННЯ ВАРТИСТЮ ТА БЮДЖЕТОМ.....	80
9.1. Процеси управління вартістю проєкту.....	80
9.2. Планування ресурсів проєкту	80
9.3. Оцінка вартості операцій.....	83
9.4. Розробка бюджету проєкту	85
9.5. Аналіз ресурсів проєкту.....	87
9.6. Контроль бюджету проєкту.....	89
Питання до розділу 9	92
РОЗДІЛ 10. УПРАВЛІННЯ ЛЮДСЬКИМИ РЕСУРСАМИ.....	93
10.1. Процеси управління людськими ресурсами проєкту	93
10.2. Формування команди проєкту	95
10.3. Процеси управління командою проєкту	98
10.4. Мотивація роботи команди проєкту.....	99
10.5. Управління конфліктами в проєктах	100
Питання до розділу 10	102
РОЗДІЛ 11. КОНТРОЛЬ ХОДУ ВИКОНАННЯ ПРОЄКТУ	103
11.1. Контроль проектної діяльності	103
11.2. Контроль виконання проєкту	103
11.3. Способи контролю ходу проєкту	105
11.4. Процеси контролю	106
11.5. Методи та види контролю проєкту.....	107
11.6. Моніторинг	108
Питання до розділу 11	111

РОЗДІЛ 12. УПРАВЛІННЯ ЗМІНАМИ, ПОСТАВКАМИ ТА ЯКІСТЮ ПРОЄКТУ	112
12.1. Управління змінами в процесі виконання проєкту	112
12.2. Управління поставками	115
12.3. Управління якістю у проєкті	116
Питання до розділу 12	121
РОЗДІЛ 13. КОРПОРАТИВНА СИСТЕМА УПРАВЛІННЯ ПРОЄКТАМИ.....	122
13.1. Корпоративна система управління проектами	122
13.2. Корпоративна методологія управління проектами	123
13.3. Реєстр проектів	125
13.4. Підтримка виконання проєкту	127
13.5. Особливості впровадження КСУП	130
Питання до розділу 13	130
РОЗДІЛ 14. МЕТОДОЛОГІЯ WATERFALL TA AGILE. SCRUM TA KANBAN.....	132
14.1. Waterfall та AGILE. Scrum та Kanban	132
14.2. Підхід до опису проектів Waterfall	132
14.3. AGILE – Гнучкі методи	135
14.4. Порівняння Scrum та Kanban	139
14.5. Універсальна схема Scrum.....	139
14.6. Можливості Kanban	143
14.7. Порівняння Kanban та Scrum. Висновки.....	146
Питання до розділу 14	146
РОЗДІЛ 15. СТАНДАРТ ISO 21500	148
15.1. Особливості стандарту ISO 21500	148
15.2. Процеси проєкту в ISO 21500	149
15.3. Учасники проектів, програм та портфелів проєктів	150
15.4. Процеси управління проєктами	154

Питання до розділу 15	158
РОЗДІЛ 16. СИСТЕМИ КЕРУВАННЯ ВЕРСІЯМА GITHUB.	159
16.1. Загальні задачі систем керування версіями	159
16.2. Що таке система контролю версій.....	159
16.3. Локальні системи контролю версій	160
16.4. Централізовані системи контролю версій.....	161
16.5. Розподілені системи контролю версій	162
16.6. Архітектура Git.....	163
16.7. Директорія Git та робоча директорія.....	167
16.8. Об'єктна модель Git.....	168
Питання до розділу 16	173
РОЗДІЛ 17. УПРАВЛІННЯ ІТ ПРОЄКТОМ В GITHUB.	174
17.1. Управління ІТ-проектом в GitHub	174
17.2. Початкові налаштування Git на локальному пристрой	174
17.3. Організація. команда. репозиторій. ролі	176
Питання до розділу 17	188
РОЗДІЛ 18. КРОКИ РЕАЛІЗАЦІЇ ІТ-ПРОЄКТІВ В GITHUB	189
18.1. Фіксація в Git.....	189
18.2. Робота з гілками	191
18.3. Кроки реалізації ІТ-проектів в GitHub	195
18.3. Робочий процес Git-flow Workflow	199
Питання до розділу 18	207
ПЕРЕЛІК ПОСИЛАЛЬ	209
СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ	211
Додаток 1	213
Додаток 2	221

ВСТУП

Знання з управління проектами передбачають розгляд множини сфер прояву життєдіяльності суспільства, що стосуються як специфіки процесів предметної області призначення проекту, так і багатьох, що доповнюють, чи суміжних сфер знань, таких як управління людськими ресурсами, управління поставками, управління змінами та конфліктами, управління комунікаціями, управління часом, управління ризиками, контроль виконання проекту та ін. Не менш важливу складову знань з управління проектами складають знання про методологію управління проектами його ініціація, паспорт проекту, планування, побудова ієрархічної структури виконання проекту та ін. Також, суттєве значення в проекті відіграє документообіг, регулюючі та регламентуючі документи та ін. І весь попередній, невичерпний перелік в умовах необхідності досягнення в проекті якості, безпеки, конкурентоспроможності в умовах дотримання зменшення витрат, врахування швидких змін, досягнення максимальної інформатизації та автоматизації та ін. при мінімальному часі на експериментування є не простою задачею реалізації різних проектів. Важливість оволодіння навичками управління проектами беззаперечна.

Матеріал навчального посібника викладено у обсязі необхідному в підготовці здобувачів ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». Посібник призначено для забезпечення проведення лекційних занять.

Ведення ІТ-проектів потребує побудови ефективних та прагматичних робочих процесів, включаючи створення та супровід якісного ПЗ, що можна застосувати в будь-якому проекті. Лише в сфері ІТ існує багато вимог – ознак якісного ПЗ: надійність, стійкість, модульність, безпечність, продуктивність, масштабованість, зручність застосування, тестування та супровід, та багато інших. При цьому проект має включати: гарну документацію, включаючи журнали змін; деталізовані узгодження та стандарти; версіонування; автоматизовані тести та ін.

Одним з зручних та ефективних засобів покращення робочих процесів ведення ІТ-проекту стали системи верифікування версій, що на сьогодні надають гнучкі інструменти, які не диктують вам як потрібно реалізувати та документувати зміни, яку стратегію злиття ІТ-проекту використовувати, на яких етапах та стадіях фіксувати результати, які інструменти застосовувати, яким стандартам commit-ів слідувати та що потрібно рецензувати. Все це зорієнтовано вашими творчими підходами.

З точки зору досягнення якості, системи контролю версій (СКВ), як системи верифікування версій, стали не від'ємною складовою проектів та продовжують швидкими темпами свій розвиток, надаючи все нових можливостей підвищення ефективності та зменшення витрат. Відповідно оволодіння засобами СКВ є невід'ємною складовою оволодіння кваліфікаційними навичками фахівців у сфері інформаційних технологій та систем.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ТА АКРОНІМІВ

БД	База даних
БПЗ	Бюджет по завершенню
ВВ	Відхилення за вартістю
ВС	Відхилення за строками
ГУП	Групи управління проектами
ЗМІ	Засоби масової інформації
ІСР	Ієрархічна структура робіт
КМП	Команда менеджменту проєкту
КП	Команда проєкту
КСТ	Корпоративний стандарт
КСУП	Корпоративна система управління проєктами
КУП	Команда управління проєктом
ОО	Освосний обсяг
ПВВР	Планова вартість виконаних робіт
ПВЗР	Планова вартість запланованих робіт
ПЗ	Програмне забезпечення
ПІБ	Прізвище Ім'я по Батькові
ПМ	Проєкт-менеджер
ПО	Плановий обсяг
ПО	Проєктний офіс
РСКВ	Розподілена система контролю версій
СДР	Структурна декомпозиція робіт
СКВ	Система контролю версій
СТП	Стандарт підприємства
ТЕО	Техніко-економічне обґрунтування
ФВ/ФС	Фактична вартість
ФВВР	Фактична вартість виконаних робіт
ЦСКВ	Централізована система контролю версій
ACWP	Actual Cost of Work Performance (AC) – Актуальна вартість виконаних робіт
AD	Active directory – активний каталог
BAC	Budget at completion – Бюджет по завершенню

BCWP	Budget Cost of Work Per formed (EV) – Бюджетна вартість виконаних робіт
BCWS	Budget Cost of Work Scheduled (PV) – Базова вартість запланованих робіт
C/SCSC	Cost/Schedule Control Systems Criteria – Вартість/Критерії системи управління графіком роботи
CI/CD	Continuous Integration/Continuous Delivery – Безперервна інтеграція/Безперервна доставка
CVS	Concurrent Versions System – централізована система управління версіями
DevOps	Development & operations — методологія автоматизації технологічних процесів збірки, налаштування та розгортання програмного забезпечення
DVCS	Distributed Version Control System - розподілена система контролю версій
EPM	Enterprise Project Management – Управління корпоративними проектами
EVM	Earned Value Management – управління заробленою вартістю
EVM/EVT	Earned Value Management/Earned Value Types – Управління заробленою вартістю/Зароблені типи цінностей
FSFS	Fast Secure File System - безпечна розподілена файлова система в просторі користувача, побудована на основі FUSE і OpenSSL
ISO	International Organization for Standardization - Міжнародна організація з стандартизації
IT	Information Technology – інформаційні технології
MS	Microsoft - Майкрософт
PMBoK	Project Management Body of Knowledge - Звід знань з управління проектами
PMI	Project Management Institute - Інститут управління проектами
PP	Project Performance - організація і контроль виконання проекту
PC	Project Controlling - Аналіз і регулювання виконання проекту
PCM	Project Change Management - Управління змінами проекту
RCS	Revision Control System - система управління редакціями
ROM-estimate	Rough Order of Magnitude Estimate – Грубі оцінки
SHA	Secure Hash Algorithm - алгоритм криптографічного хешування

SHA-1	Secure Hash Algorithm 1 — алгоритм криптографічного хешування - 1
SCCS	Source Code Control System – система контролю вихідного коду
SVN	Apache Subversion - сучасна система контролю версій. SVN – скорочення від Subversion
VCS	Version Control System - система контролю версій
WBS	Work Breakdown Structure – Ієрархічна структура робіт

РОЗДІЛ 1. ОСНОВНІ ПОНЯТТЯ. ОСОБЛИВОСТІ ІТ-ПРОЄКТІВ

1.1. Поняття «проект»

Поняття "проект" об'єднує різноманітні види діяльності, що характеризуються рядом загальних ознак, найбільш загальними з яких є наступні [1, 7, 10, 12, 13, 14]:

- спрямованість на досягнення конкретних цілей, визначених результатів;
- координоване виконання численних взаємозалежних дій;
- обмеженість у ресурсах та у часі з певним початком і кінцем.

Відмінність проекту від виробничої діяльності полягає в тому, що проект є одноразовою, нециклическою діяльністю.

Цим проект відрізняється від бізнес-процесів, дії яких можуть багаторазово повторюватись у вигляді екземплярів бізнес-процесів.

Виробничі цикли в чистому вигляді не є проектами. Однак останнім часом проектний підхід все частіше застосовується і до процесів, орієнтованим на безперервне виробництво.

Проект як система діяльності існує рівно стільки часу, скільки його потрібно для отримання кінцевого результату. Концепція проекту, однак, не суперечить концепції фірми або підприємства і цілком сумісна з нею. Більш того, проект часто стає основною формою діяльності.

Існує ряд визначень терміну "проект", кожне з яких має право на існування, в залежності від конкретного завдання, що стоїть перед фахівцем.

Ось деякі з них:

- У найзагальнішому вигляді проект (англ. - project) - це «що-небудь, що замислюється або планується», наприклад, нове програмне забезпечення для бухгалтерії.

- З точки зору системного підходу проект може розглядатися як *процес переходу з початкового стану в кінцеве* - результат за участю ряду обмежень і механізмів.

- Проект – деяка задача з певними вихідними даними і необхідними результатами (цілями), що обумовлюють спосіб її вирішення.

У сучасному розумінні проекти – це те, що змінює наш світ: розробка програмної системи, розробка програмно-технічного комплексу управління підприємством, розробка та створення літака чи пароплаву, будівництво житлового будинку або промислового об'єкту, програма науково-дослідних робіт, реконструкція

підприємства, створення нової організації, розробка нової техніки та технологій, спорудження корабля, створення кінофільму, розвиток регіону - це все проекти.

Відмінними рисами проекту є:

- *спрямованість* на досягнення конкретних цілей;
- *координоване виконання* взаємозалежних дій;
- *обмеженість у часі* з чітко визначеними початком та завершенням;
- *унікальність* дій по плануванню, виконанню робіт та результатів.

1.2. Місце управління проектами у процесі розробки програмних засобів

Місце управління проектами у процесі розробки програмного забезпечення – рис. 1.1.

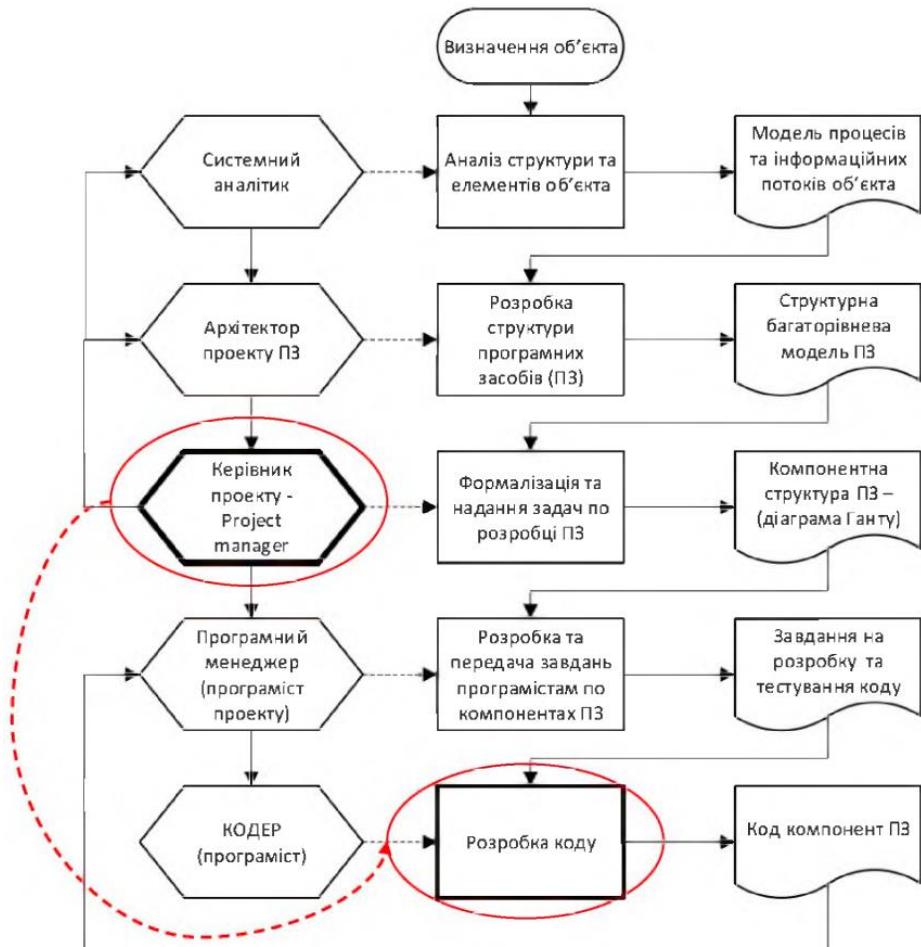


Рис. 1.1 Місце управління проектами у процесі розробки програмного забезпечення

1.3. Інвестиційний проект

Інвестиційний проект розуміється як інвестиційна акція, передбачає вкладення певної кількості ресурсів, в тому числі інтелектуальних, фінансових, матеріальних, людських, для отримання запланованого результату і досягнення певних цілей в обумовлені терміни.

Фінансовим результатом інвестиційного проекту є прибуток, дохід, матеріально-речовим результатом – нові або реконструйовані основні фонди (об'єкти) або придбання та використання фінансових інструментів, або нематеріальних активів з подальшим отриманням доходу.

В якості результатів реалізації ІТ-проекту виступають такі об'єкти як – *програмні комплекси, програмно-технічні системи, інформаційно-аналітичні системи, системи автоматизації проєктування та інші*, визначення проєкту може бути конкретизовано таким чином:

ІТ-Проєкт – цілеспрямоване, заздалегідь опрацьоване і заплановане створення або модернізація технологічних та бізнес-процесів на основі розробки програмних та програмно-технічних систем, технічної та організаційної документації для них, а також управлінських рішень і заходів щодо їх виконання.

1.4. Головні елементи проєкту в ІТ

Проект в ІТ це:

1. *План робіт* по розробці програмно-технічного забезпечення.
2. Система дій та методів *технічної підтримки виконання робіт* по розробці програмно-технічного забезпечення.
3. Система дій та методів *по забезпеченню підтримки ресурсів* (фінансових, людських, технічних та ін.) для виконання робіт по розробці програмно-технічного забезпечення.
4. *Контроль та моніторинг строків* виконання робіт відповідно до плану.
5. *Контроль дотримання якості* виконання робіт відповідно до плану.

Проект включає в себе задум (проблему), засоби його реалізації (вирішення проблеми) і одержувані в процесі реалізації результати (рис. 1.2).

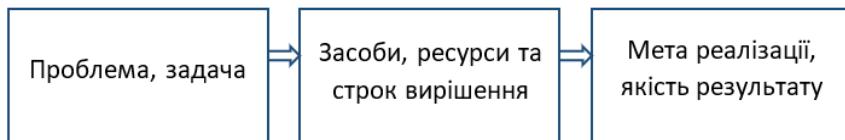


Рис. 1.2 Базова основа для розробки проєкту

Відмінні риси проекту

1. Спрямованість на досягнення окреслених цілей. Чітка постановка кінцевої мети проекту сприяє його успішній реалізації за умови правильного формулювання проміжних взаємозалежних цілей. Реалізація проекту означає послідовне досягнення цілей з найбільш низького рівня до вищого, тобто до досягнення кінцевої мети.

2. Координоване виконання взаємозалежних дій. Одні дії необхідно виконувати паралельно, інші - послідовно, і будь-яке порушення порядку їх виконання може поставити під загрозу виконання проекту взагалі.

3. Обмеженість в часі. Проекти виконують протягом певного часу (як правило, його визначають заздалегідь), по можливості більш чітко окреслюючи початок і завершення. Запорукою успішної реалізації проекту є оптимальний розподіл зусиль і ресурсів у часі, який забезпечується приведенням в порядок послідовності виконання робіт і заходів в межах проектної діяльності. На відміну від виробничої системи проект є одноразовою, а не циклічною діяльністю. Проте проектний підхід все більш часто застосовують і до безперервного виробництва. Наприклад, існують проекти виконання замовлень, де передбачені договірні терміни постачання.

4. Унікальність. Кожен проект має відмінні риси і ознаки. Не існує ідентичних проектів, навіть якщо вони передбачають виконання одинакових дій.

Троїсте обмеження

У тому випадку, коли в якості результатів реалізації проекту виступають деякі фізичні об'єкти (програмні та програмно-технічні засоби, виробничі комплекси та ін.), **визначення проекту** може бути пов'язано з певними ***обмеженнями на його розробку та виконання***.

Троїсте обмеження (рис 1.3) – це ***терміни чи час виконання, вартість робіт за проектом та зміст чи об'єм робіт проекту***. Нерідко малюють трикутник, щоб підкреслити взаємозв'язок всіх його граней. Зміна бюджету неминуче вплине на терміни і склад робіт. Зворотне справедливо і для інших граней.

У ряді галузей, таких, як авіаційна, космічна або оборонна промисловість, створювані IT об'єкти є настільки складними, що робота над ними здійснюється не в складі проектів, а в складі ***програм***, які можна визначити, як ***сукупність проектів або проект***, що відрізняється ***особливою складністю*** створюваної продукції і/або методів управління його здійсненням. При такому підході термін "проект", як правило, пов'язується з відносно короткостроковими цілями.



Рис. 1.3 Тройсте обмеження

Портфель проектів, як група проектів пов'язаних *єдиним планом та фінансуванням*.

Папка проектів (рис 1.4), як група проектів пов'язаних якими завгодно груповими признаками (географічними, джерелами фінансування, замовниками, користувачами, виконавцями, періодом часу та ін.).



Рис. 1.4 Показники проектів

1.5. Процеси проєктування та поточна операційна діяльність

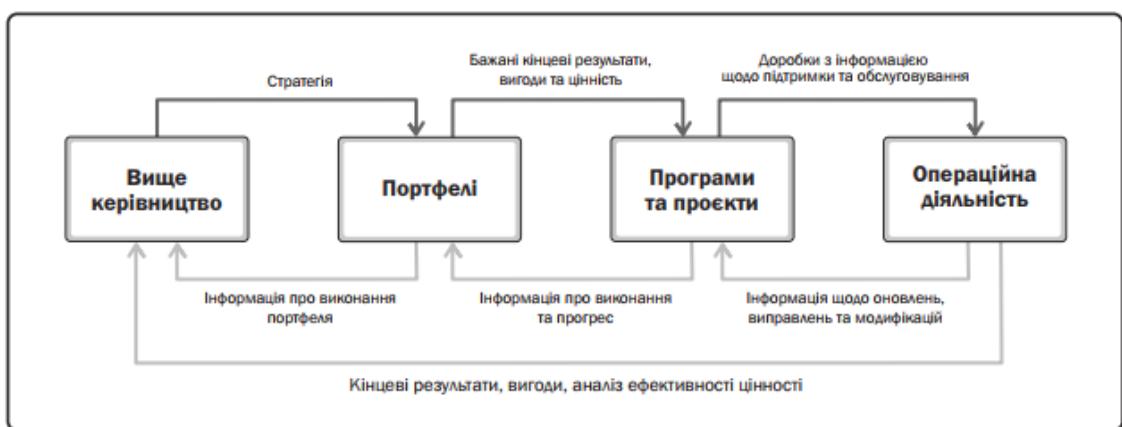


Рис. 1.5 Поточна операційна діяльність в проєктуванні процесів проєкту

1.6. Класифікація типів проектів

Типи проектів розглянуто в таблиці 1.1.

Таблиця 1.1 Типи проектів за класифікаційними ознаками

Класифікаційні ознаки	Типи проектів				
За рівнем проекту	Проект		Програма		Система
За масштабом (розміром проекту)	Малий		Середній		Мегапроект
За складністю	Простий	Організаційно складний	Технічно складний	Ресурсно складний	Комплексно складний
За строками реалізації	Коротко-строковий		Середній		Мегапроект
За вимогами до якості та засобами її забезпечення	Бездефектний		Модульний		Стандартний
За вимогами до обмежуваності ресурсів сукупності проектів	Мультипроект		Монопроект		
За характером проекту / рівнем учасників	Міжнародний (сумісний)		Вітчизняний Державний Територіальний Місцевий		
За характером цільової задачі проекту	Антикризовий		Реформування / реструктуризація		
За об'єктом інвестиційної діяльності	Маркетинговий Освітній Фінансовий Інвестиційний		Інноваційний Надзвичайний Реальний Інвестиційний		
За головною причиною виникнення проекту	Можливість, що з'явилася		Необхідність структурно-функціональних перетворень	Реорганізація	
	Надзвичайні			Реструктуризація	

Малі проекти невеликі за масштабом, прості і обмежені за обсягами.

Так, в американській практиці малі проекти:

- капіталовкладення: до 1 млн. дол.;

- трудовитрати: до 3-4 тис. людино годин. (5 виконавців ~ 3 місяців).

Приклади типових малих проєктів: промислові програмні системи, модернізація діючих програмних чи програмно-технічних систем.

Малі проєкти допускають ряд спрощень у процедурі проектування і реалізації, формування команди проєкту (можна просто короткочасно перерозподілити інтелектуальні, трудові та матеріальні ресурси).

Разом з тим, існує **ускладненість в управління допущених помилок, в зв'язку з дефіцитом часу** на їх усунення, що вимагає досить ретельного визначення об'ємних характеристик проєкту, учасників проєкту та методів їх роботи, графіка проєкту і форм звіту, а також умов контракту.

Мега-проєкти – це **цильові програми**, що містять **безліч взаємопов'язаних проєктів, об'єднаних спільною метою**, виділеними ресурсами і відпущені на їх виконання часом. Такі програми можуть бути **міжнародними, державними, національними, регіональними** (наприклад, розвиток вільних економічних зон, малих народностей і т.ін.), **міжгалузевими** (зачіпати інтереси декількох галузей економіки), галузевими і змішаними. Як правило, програми **формуються, підтримуються і координуються на верхніх рівнях управління**: державному (міждержавному), республіканському, обласному і т.ін.

Мега-проєкти мають ряд відмінних рис:

- **високою вартістю** (більш 100 млн. дол. і більше);
- **капіталоємністю** - потреба у фінансових коштах в таких проєктах, як правило, вимагає нетрадиційних (акціонерних, змішаних) форм фінансування, зазвичай силами консорціуму фірм;
- **три阀алістю** реалізації (до 3-х років і більше);
- **необхідністю** участі інших країн;
- **віддаленістю** районів реалізації (**розгалуженістю**), а отже, додатковими витратами на інфраструктуру;
- **впливом на** соціальну та економічну структуру **середовища регіону і навіть країни** в цілому (наприклад 5G, Е-держава, Дія та ін.).

Найбільш характерні приклади галузевих мегапроєктів – це проєкти, що виконуються для всієї країни, наприклад, проєкти **державних інформаційно-аналітичних систем**.

Складні проєкти мають на увазі наявність технічних, організаційних або ресурсних завдань, вирішення яких передбачає **нетривіальні підходи і підвищені витрати на їх рішення**. З використанням складних технічних засобів та обладнання

для виконання робіт і використання результатів розробки. Наприклад тренажерні системи управління атомною станцією.

Короткострокові проекти зазвичай реалізуються на практиці як модернізація чи впровадження вже існуючих продуктів.

На таких об'єктах замовник звичайно йде на збільшення остаточної (фактичної) вартості проекту проти початкової, оскільки найбільше він зацікавлений у якнайшвидшому його завершенні.

Бездефектні проекти в якості домінуючого чинника використовують підвищену якість. Зазвичай вартість бездефектних проектів дуже висока і вимірюється сотнями мільйонів доларів, наприклад атомні електростанції.

Міжнародні проекти зазвичай відрізняються значною складністю виконання та координації робіт виконавців, значною вартістю та значними ризиками своєчасного виконання у рамках визначеного бюджету проекту.

1.7. Компоненти проекту

Продукт – результат (або набір результатів) розробки за контрактом. Продукт – це те, що хоче отримати замовник.

Проект – це *опис процесу* створення програмно-технічного продукту. Це те, що робить команда, щоб видати замовнику продукт. Замовнику, зазвичай, проект не цікавий.

Будь-який **проект** розроблюється з метою *отримати продукт*.

Замовник згоден *заплатити і почекати*. Причому і перше і друге – в обмеженій кількості.

В обмін на це замовник хоче, щоб продукт повністю *відповідав його очікуванням*.

Замовник не обов'язково правильно сформулює очікування як слід – це турбота *аналітика, архітектора проекту та керівника проекту*.

Основна *задача керівна* проекту дати замовнику те, *що він хоче*, не запрошуючи у нього *додаткового часу або грошей* (гроші можуть бути виражені в ресурсах).

1.8. Блоки робіт Project Management

1. Управління часом (*ми повинні вкластися в термін*).
2. Управління вартістю (*виконавці не повинні перевищити бюджет*).

3. Управління змістом (потрібно уточнити бажання замовника і правильно їх реалізувати).
4. Управління якістю (контроль відповідності результатів завданню).
5. Управління ризиками (контроль повноти та своєчасності виконання у рамках бюджету).
6. Управління закупівлями (якщо ми використовуємо субпідрядників).
7. Управління персоналом (підбір компетентних виконавців задач).
8. Управління комунікаціями (можливості, повноваження та засоби спілкування та синхронізації дій виконавців, керівників та зацікавлених осіб).
9. Управління інтеграцією (об'єднати різні інтереси, дії і результати для досягнення цілей проєкту)

Питання до розділу 1

1. Що таке проєкт з системної точки зору?
2. Які відмінні риси проєкту?
3. Які є типи проєкту?
4. Які є основні структури управління проєктами?
5. Назвіть основні структури управління проєктами?
6. Компоненти проєкту?
7. Яке займає місце управління проєктом у процесі розробки програмних заходів?

РОЗДІЛ 2. ОСНОВНІ ПОНЯТТЯ. ЖИТТЄВИЙ ЦИКЛ. МОДЕЛІ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ

2.1. Життєвий цикл проєкту

Початок проєкту прийнято називати *ініціацією*, а *закінчення – закриттям*.

Між цими двома подіями розташовуються не лінійно *групи робіт* [1, 7, 13]:

- *Планування;*
- *виконання робіт;*
- *моніторинг;*
- *управління.*

Нелінійність в тому, що дані *процеси послідовні, але ітеративна* (тобто повторюються).

Спланований проєкт починає виконуватися і відслідковуватися, проте у міру виконання робіт відстеження виявляє накопичилися *потреби в змінах*. *Початкові плани коригуються*, розробка та подальший моніторинг ведеться вже за ним.

2.2. Стадії життєвого циклу проєкту

З точки зору фінансування проєкту *життєвий цикл проєкту* можна *розділити на чотири основні смыслові стадії*:

- *Передінвестиційна.*
- *Інвестиційна.*
- *Експлуатаційна.*
- *Ліквідаційна...*

На *передінвестиційній стадії* проводяться:

- *аналіз інвестиційних можливостей;*
- *попереднє TEO;*
- *планування;*
- *організація фінансування.*

На *інвестиційній стадії*:

- *переговори і укладення контрактів;*
- *проектування;*
- *розробка програмно-технічних систем;*
- *тестування;*
- *наповнення реальними даними;*

- навчання користувачів програмних систем.

На стадії експлуатації:

- приймання і запуск;
- використання програмно-технічного забезпечення;
- заміна програмно-технічного забезпечення;
- розширення, модернізація, інновація.

На стадії ліквідації проєкту проводиться завершення проєкту як одноразового заходу.

2.3. Основні фази життєвого циклу проєкту

Для ІТ-проєктів зазвичай виділяють такі фази в роботі менеджера проєкту:

- **концептуальну фазу**, що включає формульовання цілей, аналіз інвестиційних можливостей, обґрунтування здійсненості (техніко-економічне обґрунтування) і планування проєкту;

- **фазу розробки проєкту**, що включає визначення структури робіт і виконавців, побудова календарних графіків робіт, бюджету проєкту, розробку проєктно-кошторисної документації, переговори і укладення контрактів з підрядниками і постачальниками;

- **фазу виконання проєкту**, що включає роботи по реалізації проєкту, в тому числі розробка програмних засобів, закупка технічних засобів (серверів, комп'ютерів, мережевого обладнання), тестування програмно-технічних комплексів, навчання персоналу;

- **фазу завершення проєкту**, що включає в загальному випадку приймальні випробування, дослідну експлуатацію і здачу проєкту в експлуатацію;

- **експлуатаційну фазу**, що включає приймання і запуск, заміну обладнання, розширення, модернізацію, інновацію.

Структури управління проектами.

Існує кілька типів структур, які широко застосовуються в управлінні проектами: функціональна, матрична та проектна [1, 7].

Функціональна структура управління. При такій структурі управління здійснює лінійний керівник через групу підлеглих йому функціональних керівників, кожний з яких керує певними підрозділами в межах доручених функцій (рис.2.1).

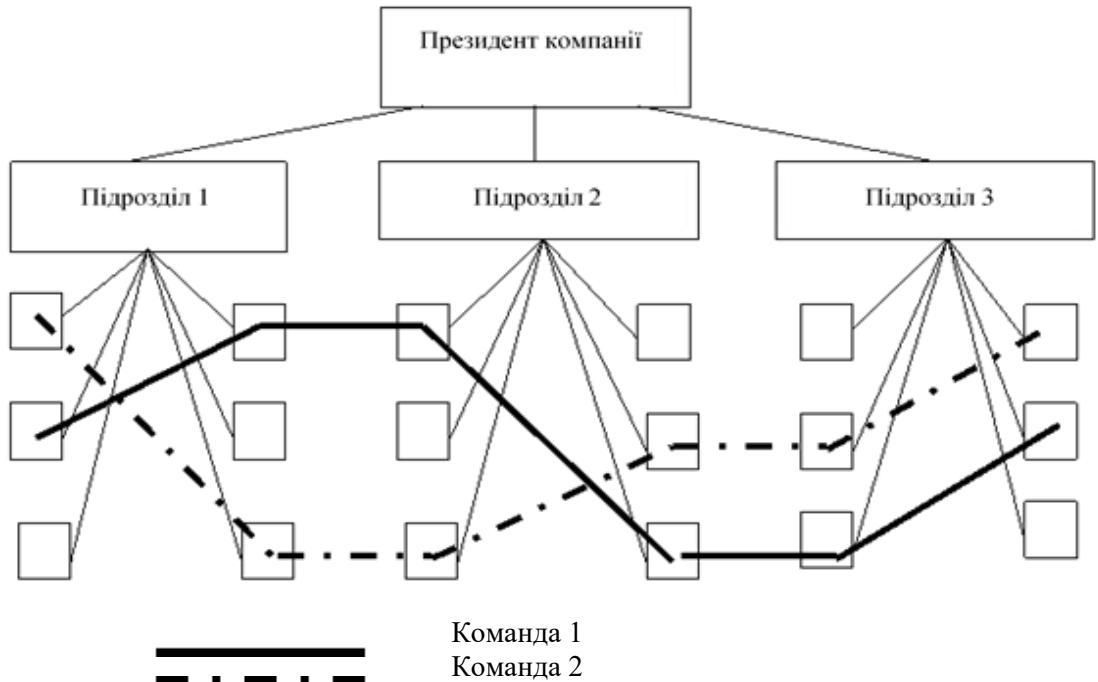


Рис. 2.1 Функціонально організаційна структура

Матрична *структурата управління* створюється на базі функціональної. В цьому випадку взаємини базуються на прямих вертикальних зв'язках "керівник - підлеглий". З метою вирішення конкретних проблем створюються тимчасові проектні групи, які очолюють керівники проектів.

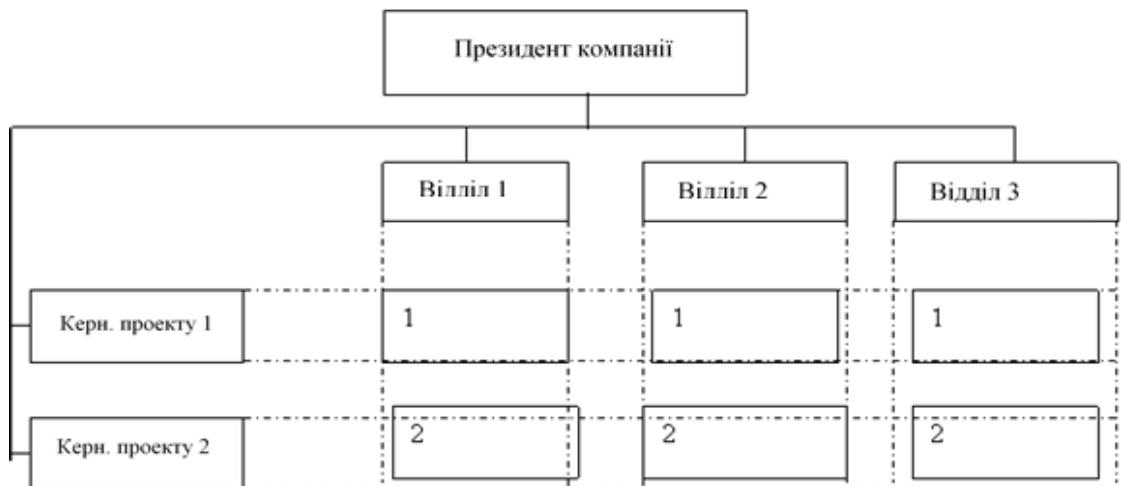


Рис. 2.2 Матрична структура управління

Ці групи формують з фахівців відповідних функціональних відділів, які знаходяться на різних рівнях ієрархії управління. Керівники проектів взаємодіють з функціональними відділами по горизонталі; ці зв'язки накладаються на традиційні вертикальні зв'язки "керівник - підлеглий", утворюючи матрицю взаємодії (рис. 2.2).

Матрична структура управління уможливлює гнучке маневрування людськими ресурсами завдяки перерозподілу їх між проектами. Для того щоб ця структура була ефективною, необхідно мати ефективну систему контролю за виконанням робіт, пов'язаних з проектом, якістю їх виконання, витратами і термінами. Необхідно постійно стежити за тим, щоб фактичні показники відповідали плановим.

Керівник проекту повинен мати у своєму розпорядженні докладну інформацію про стан виконання проекту в загальному вигляді, а керівники відділів - про роботи, які виконують їх відділи. На базі цих даних складають звіти, які керівники проектів обговорюють з підлеглими їм групами. Такі обговорення можуть відбуватися щотижня, а при критичних ситуаціях - кожен день.

Матричну структуру управління доцільно застосовувати при реалізації малих і середніх проектів. Для великих проектів така структура малоєфективна, оскільки при цьому різко підвищується складність мережі комунікацій, а це призводить до істотного уповільнення процесів прийняття управлінських рішень.

Проектна структура управління. При вирішенні проблемних завдань, пов'язаних з переорієнтацією цілей організації або зміною шляхів їх досягнення, найбільш ефективною формою реалізації проектів є проектне управління - сукупне управління трудовими, фінансовими, матеріальними та енергетичними ресурсами, необхідними для забезпечення реалізації проекту в обумовлений термін в межах запланованої кошторисної вартості і з відповідною якістю (рис. 2.3).

У проектній організаційній формі управління реалізуються вимоги системного підходу до управління, відповідно до якого роботи, які забезпечують розв'язання визначененої проблеми або досягнення кінцевої мети, розглядаються з позицій не постійної ієрархії підпорядкування, а саме досягнення певної мети або вирішення певної проблеми. Для управління розробкою конкретних проектів і програм створюються комплексні органи з відповідними повноваженнями. Вони покликані забезпечити пріоритет загальних, глобальних цілей організації над приватними, локальними цілями функціонального характеру; підвищити відповідальність за кінцевий результат робіт; децентралізувати вирішення оперативних завдань, забезпечивши гнучке і оперативне реагування на зміни зовнішніх і внутрішніх умов.



Рис. 2.3 Проектна структура управління проектами

Проектне управління як організаційна форма спочатку мало вид тимчасового структурного утворення, яке застосовувалося в межах діючої лінійно-функціональної структури управління. Тривалість життєвого циклу такої організаційної форми залежала від часу, протягом якого організація досягала поставлених нею цілей і завдань. В процесі функціонування цієї організаційної форми почав формуватися спеціальний організаційний механізм, тобто якісно нова схема взаємодії підрозділів і окремих виконавців. Це послужило причиною необхідності науково-методичного обґрунтування нової організаційної форми проектного управління.

Відповідно до проектної структури управління для вирішення конкретного завдання, наприклад проектування і будівництва об'єкта, на підприємстві створюють спеціальну робочу групу, яку після реалізації проекту розпускають. При цьому залучений до робочої групи персонал і ресурси повертаються до відповідних спеціалізованих підрозділах. Для вирішення завдань перспективного розвитку на підприємстві створюють спеціальний підрозділ, який вирішує виключно питання стратегії, а керівники проектів зосереджують свою увагу на виконанні конкретних завдань.

Однією з важливих проблем, які виникають в організаційних структурах, побудованих за принципом проектного управління, є розподіл функцій між так званими проектними та організаційними рівнями управління. Іншими словами, потрібно вирішувати, яку частину управління центр може передати на нижчий проектний рівень, а виконання яких функцій залишити на верхньому рівні. Конфліктні ситуації між центром і проектною групою виникають здебільшого через наявність питань, за вирішення яких відповідають обидва рівня управління, а також з-за не визначеності, яка потребує прийняття рішень як на організаційному, так і на проектному рівнях управління.

На проектному рівні готують проектні рішення для подальшої передачі їх на організаційний рівень управління.

На організаційному рівні вибирають проекти, визначають терміни завершення їх розробки і реалізації і розподіляють ресурси між проектами. Вибір проектів і визначення термінів завершення їх розробки є стратегічними завданнями, рішення яких вимагає великого обсягу знань з багатьох областей - техніки, економіки, соціології та ін.

Зазначені рівні управління взаємодіють шляхом передачі зверху вниз інформації інструктивного характеру, а від низу до верху - поточних даних про проект. При цьому на організаційний рівень передається в достатній мірі агрегована інформація. З метою перевірки активності роботи проектних груп їх періодично оцінюють, найбільш часто після завершення чергового етапу проекту.

Однією з важливих проблем, від вирішення якої залежить ефективність проектного управління в загальному, є оцінка діяльності функціонального виконавця в системі проектного управління. Здебільшого такі системи характеризуються тим, що функціональний виконавець підпорядковується щонайменше двом керівникам: функціональному керівнику та керівнику проекту. Первому з них виконавець підпорядкований постійно, другому - тимчасово, на період виконання робіт, пов'язаних з реалізацією проекту. Часто виконавець одночасно бере участь в декількох проектах, а тому може підкорятися відразу кільком керівникам. Взагалі проблема оцінки результату діяльності і потенціалу окремих виконавців дуже складна. Здебільшого вона виникає тоді, коли проект завершується або керівник проекту збирається підвищити працівника по службі.

Системи проектного управління, які орієнтуються на кінцеву мету – виконання проекту, сприяють скороченню термінів його виконання, підвищення рівня оперативності вирішення поточних питань, пов'язаних з виконанням проекту, більш збалансованому погодженням програми робіт з ресурсними можливостями підрядної

організації; економії ресурсів, а також більш об'єктивній оцінці діяльності окремих виконавців.

Можна зазначити, що системи проектного управління мають певні недоліки. Організаційна структура є найбільш важливим механізмом управління проектом. Вона дає можливість реалізовувати всю сукупність функцій, процесів і операцій, необхідних для досягнення поставлених перед проектом цілей.

Питання до розділу 2

1. Що таке життєвий цикл проєкту?
2. Які стадії життєвого циклу виділяють у проєкті?
3. Які фази життєвого циклу виділяють у проєкті?
4. Які основні моделі та методи управління проєктами ви знаєте?
5. Які є типи проєктів?
6. Які є основні структури управлінні проєктами?
7. Назвіть основні елементи проєкту?

РОЗДІЛ 3. ПЛАНУВАННЯ ПРОЄКТУ. ПАСПОРТ ПРОЄКТУ

3.1. Ініціація проєкту

На передінвестиційній стадії:

- проводиться аналіз інвестиційних можливостей;
- готується попереднє ТЕО;
- проводиться планування робіт, строків, вартості, виконавців;
- визначаються та організуються процедури фінансування.

3.2. Спонсор проєкту

Спонсор проєкту - це фігура, яка забезпечує проєкт фінансовими ресурсами. Це найважливіший учасник на етапі ініціації проєкту.

Спонсором є той, хто стверджує вартість проєкту (а також що за цю вартість варто зробити).

Спонсором може виступати сам замовник (якщо нам доручили безпосередньо домовлятися з ним про ціну, з поправкою на очікуваний прибуток компанії).

Нерідко спонсором виступає **вищий або середній менеджмент організації замовника** або **виконавця проєкту** (наприклад, коли проект вже підготовлений та контракт на його виконання заключений замовником, а директор лише доручає його виконати співробітникам, оголошуючи про доступне фінансування і терміни).

3.3. Затвердження проєкту

По-перше – обмеження. У фазі ініціації поки ще нічого не знаємо про проєкт (можливо і спонсор має лише хитке уявлення про те, чого слід досягти).

По-друге – попередні оцінки. У фазі ініціації можливо тільки високо рівневі планування, «грубі оцінки».

- Використовують досить грубі оцінки (*ROM-estimate, Rough Order of Magnitude Estimate - Грубий порядок оцінки величини*) - припускають відхилення до 50%.
- Важливо зафіксувати можливість помилок у оцінках та їх діапазон.

По-третє – досяжність поставленої мети. Менеджер проєкту відповідає за проєкт в цілому. Це фраза не тільки красиво звучить, вона також наділяє відповідальністю та повноваженнями.

По-четверте – формалізація домовленостей. Найважливішим виходом групи процесів ініціації є «**паспорт проєкту**».

3.4. Формування паспорт проєкту

Паспорт проєкту – документ, що формалізує домовленості зі спонсором в ході ініціації проєкту. Формування паспорту – це вже прояв методологічної специфіки управління проєктами [1, 7].

Для ІТ-проєктів, це **один з важливіших документів**, який разом з контрактом формалізує **основні особливості виконання проєкту ІТ**.

Паспорт проєкту не регламентує відносин між організацією виконавця і замовником. Паспорт проєкту визначає домовленості виконавців проєкту (менеджера проєкту, керівника проєкту) зі спонсором проєкту та замовником.

Головна властивість паспорту ІТ-проєкту – це його **незмінність**. Це найстабільніший документ проєкту (саме тому, що він задає базові рамки). Нерідко, коли у спонсора або у команді виникає необхідність змінити щось у паспорті проєкту, то приймається рішення просто закрити поточний проєкт і розпочати новий (з новим паспортом).

Важливість паспорту проєкту

Причина перша: паспорт наділяє повноваженнями менеджера проєкту.

Саме завдяки паспорт у менеджер проєкту може вимагати собі на проєкт потрібну кількість розробників, тестувальників, аналітиків і так далі, а, найчастіше, ще й наполягати на певній кваліфікації кадрів. Це вкрай важливо, якщо у вашій організації виконується безліч проєктів одночасно і, скажімо, начальник відділу розробки зовсім не зацікавлений віддати свого кращого програміста вам на 5-6 місяців.

Причина друга: паспорт фіксує умови і обмеження проєкту.

Умови паспорту не повинні збігатися з умовами контракту (який компанія укладе з замовником), або, не суперечити їм. Якщо через кілька місяців спонсор-директор, від імені компанії уклав із замовником **додатковий договір** (що розширює рамки проєкту), то нові домовленості можуть стати предметом **нового проєкту** (можливо «нового проєкту», якщо менеджер проєкту погодиться в ньому брати участь). Паспорт дозволяє відокремити зобов'язання за проєктом від зобов'язань компанії за контрактом.

В одному контракті можуть бути декілька проєктів и декілька різних паспортів.

Підготовка паспорту проєкту

Паспорт проєкту **створюється менеджером** проєкту і **затверджується спонсором.**

Керівник проєкту максимально зацікавлений в паспорт, а з ним і вся команда, якою він керує.

Склад паспорту проєкту:

1. ***Мета проєкту.***
2. ***Дані*** (ПІБ, тел., e-mail та інше) менеджера проєкту, спонсору проєкту. Опис організацій, що приймають участь.
3. ***Троїсте обмеження:***
 - Терміни загальні і найважливіших етапів.
 - Бюджет загальний і поетапний.
 - Зміст робіт по проєкту (за основними етапами).
4. ***Зацікавлені особи*** (самі ключові – як мінімум користувачі субпідрядники та інші).
5. Очікувані результати для спонсора, замовників і користувачів проєкту.
6. Етапи проєкту (5-10 етапів – задач верхнього рівня).
7. Пов’язані проєкти та роботи.
8. Перелік функцій системи, що розроблюється. Показники споживчих якостей.
9. Індикатори досягнення мети. Кількісні характеристики можливостей системи, що розроблюється.
10. Ресурси, необхідні для реалізації проєкту. Перелік необхідних ресурсів та їх кількості.
11. Ризики проєкту, їх наслідки та шляхи їх усунення.

Планування проєкту починається задовго до його початку.

Зацікавлені особи проєкту - всі люди, інтереси яких зачіпає реалізація проєкту (позитивним або негативним чином).

Надзвичайно корисний розділ паспорт проєкту - «**що не є**» вимогою до продукту.

Зацікавлені сторони проєкту (Project Stakeholders)

Найбільш загальна угруповання зацікавлених сторін (рис 3.1):

- 1) *Представники цільової групи (ti, хто буде користуватися продуктом).*
- 2) *Потенційні партнери (з чисю допомогою може здійснюватися проєкт).*
- 3) *Опоненти (зацікавлені в тому, щоб проєкт не здійснився).*
- 4) *Виконавці проєкту (зацікавлені завершити проєкт вчасно і якісно).*

- 5) Замовники проекту (зацікавлені отримати прибуток).
- 6) Спостерігачі.
- 7) Вищі організації та інстанції.
- 8) Держава (зацікавлене в дотриманні законодавчих і нормативно-правових актів).
- 9) ЗМІ.
- 10) Громадськість.

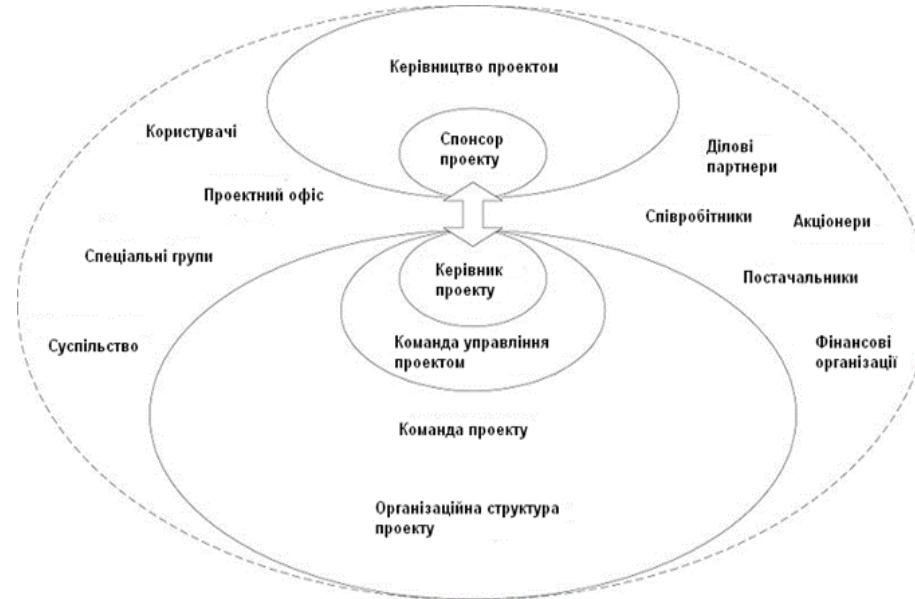


Рис. 3.1 Зацікавлені сторони проєкту

Специфіка фази ініціації в тому, що рішення даються легко, але коштують дорого. Перерахуємо деякі можливі помилки менеджера проєкту в цій фазі ініціації:

– **Необхідність додаткової інформації щодо оцінок.** Менеджер проєкту видає попередні оцінки, ґрунтуючись виключно на власному досвіді і інтуїції. Однак якщо у менеджера проєкту є найменший шанс скористатися чиєюсь експертizoю — обов'язково слід звернутися за допомогою.

– Можливість перебільшення оцінок (padding). Як би не була складна оцінка - не можна перестраховуватися, завищуючи її. Діапазон коливання (+/- 50%) цілком допустимо. Якщо діапазон виходить більше - потрібно підвищити точність оцінок, або потрібно розбити проєкт на кілька під проєктів.

Результат роботи по проєкту зі спонсором на етапі ініціації:

- спонсором затверджено керівника проєкту;
- документально затверджено запуск проєкту;
- документально затверджено паспорт проєкту.

Основні групи задач (етапів) програмно-технічних проектів

1. Збір та аналіз інформації про особливості системи, що проектується.
2. Розробка концептуальної моделі та архітектури програмних та технічних засобів.
3. Формування технічного завдання та оцінка вартості.
4. Розробка плану виконання робіт та фінансування.
5. Визначення виконавців, підрядників та постачальників.
6. Закупівля необхідного для виконання робіт.
7. Розробка програмних засобів (БД, коду для серверної частини, клієнтської частини, мережі, гаджетів ...).
8. Поставка необхідного обладнання для користувачів системи.
9. Тестування (локальне, комплексне та тех. засобах користувачів).
10. Підготовка документації та навчання користувачів.
11. Впровадження (наповнення системи даними користувачів, програма випробувань, здача-приймання робіт ...).
12. Супроводження робіт на етапі дослідного впровадження.

Питання до розділу 3

1. Що таке ініціація проєкту у проєктному управлінні?
2. Які процедури включає ініціація проєкту?
3. Що є ціллю і завданням проєкту?
4. Що таке паспорт проєкту?
5. Які основні розділи паспорту проєкту?
6. Зацікавлені сторони та учасники проєкту?
7. Складності у підготовці паспорту проєкту?
8. Результати роботи по проєкту зі спонсором ?

РОЗДІЛ 4. ПЛANI ПРОЄКТУ. СТРУКТУРНА ДЕКОМПОЗИЦІЯ РОБІТ

4.1. Планування проєкту

Планування проєкту - це процес формування рішень, що визначають порядок, в якому повинна відбуватися послідовність окремих заходів, дій та робіт за проєктом.

Планування займає основне місце в управлінні проєктом, будучи організуючим початком всього процесу щодо його використання [1, 7, 11].

Поняття «план» має багато значень, в нього часто вкладається різний зміст.

Частіше поняття «план проєкту» відноситься до *плану дій на етапах розробки програмно-технічних засобів*. Але для значних, більш-менш великих проєктів поняття «план проєкту» може відноситься до всіх етапів, включаючи підготовчі етапи.



Рис. 4.1 Планування у проєкті

В управлінні проєктом планування (рис. 4.1) є організаційним початком процесу реалізації проєкту.

Сутність планування проєкту полягає в обґрунтуванні цілей і засобів їх досягнення на основі:

- визначення переліку усіх типів робіт;
- виявлення усіх типів ресурсів проєкту;
- обрання ефективних методів і засобів виконання робіт;
- встановлення взаємодії учасників та організацій, комунікацій проєкту.

Процес розробки планів охоплює більшість основних етапів проєктного циклу:

- створення концепції проєкту;
- вибір стратегічного рішення щодо виконання проєкту; розробка деталей проєкту;
- розробка контрактних пропозицій;

- заключення контрактів;
- виконання робіт з розробки програмно-технічних засобів та тестування результатів розробки;
- передача результатів замовнику та завершення проєкту.

На етапі планування проєкту визначають всі необхідні параметри його реалізації:

- тривалість (взагалі і за окремими роботами);
- потреба в трудових, матеріально-технічних і фінансових ресурсах;
- терміни поставки обладнанням, програмних засобів і послуг;
- терміни розробки і тестування програмних продуктів;
- залучення до проєкту зовнішніх організацій;
- терміни навчання користувачів і підготовки документації.

Прийняті рішення щодо цих параметрів повинні забезпечити реалізацію проєкту в задані терміни з мінімальними витратами ресурсів і високою якістю виконання робіт.

План управління проєктом — це сукупність всіх проєктних планів.

Можливий склад плану управління проєктом:

- План управління змістом.
- План управління часом [3].
- План управління вартістю.
- План управління якістю.
- План управління персоналом.
- План управління комунікаціями.
- План управління ризиками.
- План управління поставками.
- План по вівах.
- План управління змінами.

4.2. Основні роботи менеджера проєкту по плануванню

1. *Планування змісту проєкту і його документування.*
2. *Опис змісту проєкту*, визначення основних етапів реалізації проєкту, декомпозицію їх на більш дрібні і керовані елементи.
3. *Складання кошторису*, оцінка вартості ресурсів, необхідних для виконання робіт проєкту.

4. *Визначення робіт*, формування списку конкретних робіт, які забезпечують досягнення цілей проєкту.
5. *Визначення послідовності робіт*, визначення і документування обмежень на роботи.
6. Проведення оцінки тривалості робіт, трудовитрат і інших ресурсів, необхідних для виконання окремих робіт.
7. Проведення розрахунку тривалості робіт і вимог до ресурсів.
8. Планування ресурсів, визначення того, які ресурси і в яких кількостях потрібні для виконання робіт проєкту, визначення термінів виконання робіт з урахуванням обмеженості ресурсів.
9. Складання бюджету, прив'язка кошторисних витрат до конкретних видів діяльності.
10. Створення плану-графіку проєкту, збір результатів інших процесів планування і їх об'єднання в загальний документ.

Допоміжні роботи при плануванні проєкту:

1. *Планування якості*, визначення стандартів якості, відповідних даним проєктом, і пошук шляхів їх досягнення.
2. *Формування команди проєкту* на всіх стадіях життєвого циклу проєкту, набір необхідних людських ресурсів, включених в проєкт що працюють в ньому.
3. *Розподіл проєктних ролей*, відповідальності і відносин підлегlostі.
4. *Планування поставок та субпідрядників*, визначення того, яким чином, коли і за допомогою кого закуповувати і поставляти.
5. *Планування комунікацій*, визначення інформаційних і комунікаційних потреб учасників проєкту: кому і яка інформація необхідна, коли і як вона їм повинна бути доставлена.
6. *Ідентифікацію та оцінку ризиків*, визначення того, який фактор невизначеності і в якій мірі може вплинути на хід реалізації проєкту, визначення сприятливого і несприятливого сценарію реалізації проєкту, документування ризиків.

4.3. Структурна декомпозиції робіт (СДР- WBS)

Структурна декомпозиція робіт ((СДР) - Work Breakdown Structure, (WBS – рис. 4.2) [2], іноді Ієрархічна структура робіт (ICP) або Структура розбиття робіт (CPP)) — ієрархічна структура послідовної декомпозиції проєкту на під проєкти, пакети робіт різного рівня, пакети детальних робіт.

СДР є базовим засобом для створення системи управління проєктом і дозволяє:

- *вирішувати проблеми* організації робіт,
- *розподілу відповідальності*,
- *оцінки вартості*, створення *системи звітності*,
- ефективно підтримувати процедури *збору інформації про виконання робіт*,
- *відоображення результатами* для узагальнення графіків робіт, вартості, ресурсів і дат завершення.

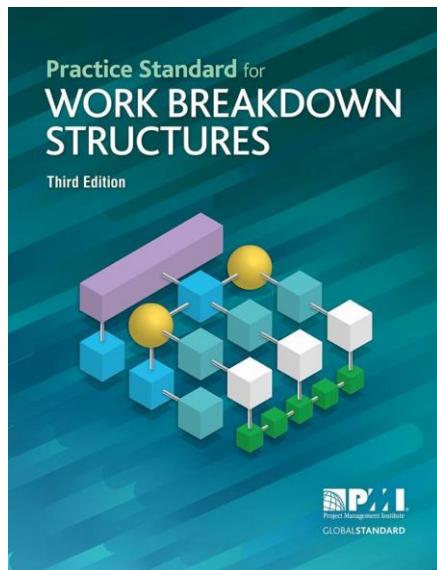


Рис. 4.2 Стандарт WBS

Етапи розробки WBS:

- 1) визначення ступеня деталізації** проєктних робіт (так, щоб вони піддавались оцінці);
- 2) визначення кількості рівнів** (як правило три-чотири, для сучасних компаній - чотири оптимально);
- 3) розробка структури кожного рівня** (формуються горизонтальні рівні);
- 4) опис елементів WBS** (стисла назва кожної складової WBS);
- 5) формування системи кодування** (кодуються всі блоки);
- 6) проведення зворотних обчислень** (розраховуються витрати по кожному блоку “знизу до гори” за принципом: відділ локалізації – субпідрядник).

Для одного і того самого проєкту можна створити кілька WBS із різною кількістю рівнів та елементів на кожному рівні залежно від принципу, який

покладається в основу розбивки проекту на його складові. **Принципи формування структури WBS**

Принципи формування структури WBS:

1. **за продуктами або підпроектами** (підпроект 1, підпроект 2, підпроект 3, ...);
2. **за фазами проекту** (проектування, розробка програм, тестування, приймання і таке інше);
3. **за місцем виконання робіт** (серверна частина, мережева частина і таке інше);
4. **за центрами витрат** (компанія 1, компанія 2, компанія 3, ...).

За дотримання будь-якого з цих принципів WBS – це поділ проекту на його складові елементи на логічній основі.

Для створення WBS структуризація може провадитися по таких рівнях:

- **рівень 1 – проект;**
- **рівень 2 – стадії або підпроекти;**
- **рівень 3 – системи або блоки;**
- **рівень 4 – робочі задачі.**

Приклад три-рівневої робочої структури проекту зі створення комп'ютерного центру в організації (рис 4.3). Перший рівень – це сам проект, другий – це під-проекти, сформовані за продуктовим принципом: забезпечення кадрами, технічне забезпечення, програмне забезпечення і управління проектом.



Рис. 4.3 Створення комп'ютерного центру

На третьому рівні WBS перебувають робочі пакети для перших трьох підпроєктів, а управління проєктом не деталізується. Тобто слід підкреслити, що глибина розбивки за певними блоками може бути різною.

4.4. Робочий пакет

На найнижчому рівні робочої структури кожного проєкту знаходиться *робочий пакет* (*work package*). Він являє собою групу робіт чи операцій, які піддаються оцінці з погляду визначення витрат і наділення ресурсами, тривалості виконання та призначення відповідального.

Кожен робочий пакет повинен мати чисельну інформацію за наступними характеристиками:

- *обсяг і перелік робіт, які треба виконати*: у певній послідовності, з встановленням взаємозв'язків між роботами;
- *відповідальні за виконання робочого пакету*: перелік посад, та/чи функціональні обов'язки виконавців пакету;
- *бюджет*: розрахунок витрат на виконання робіт, придбання необхідних ресурсів, оплата послуг сторонніх установ та праці виконавцям робочого пакету, накладні витрати;
- *потребні ресурси*: перелік необхідних матеріальних, трудових, інформаційних, адміністративних ресурсів, засобів забезпечення (приміщення, транспорт, засоби зв'язку, тощо);
- *дати початку і кінця*: встановлюються календарні дати виконання, чи вказується бажаний загальний термін, необхідний для виконання робочого пакету.

СДР-WBS є зручним засобом управління для *менеджера проєкту*, так як дозволяє:

- *визначити роботи*, пакети робіт, що забезпечують досягнення проєкту;
- *створити* зручну, відповідну цілям проєкту **структуру звітності**;
- *визначити* на відповідному рівні деталізації плану **віхи** (ключові результати), які повинністати контрольними точками по проєкту;

– розподілити відповідальність за досягнення цілей проєкту між його **призначеними виконавцями** і тим самим гарантувати, що всі роботи по проєкту мають відповідальних і не випадуть з поля зору.

Можливий алгоритм планування ІТ-проєкту:

1. Визначити, як буде будуватися планування.
2. Зібрати і оцінити вимоги.
3. Сформувати концепцію (scope).
4. Прийняти рішення про закупки.
5. Визначити команду.
6. Створити ICP (ієрархічну структуру робіт) (WBS).
7. Оцінити необхідні ресурси.
8. Оцінити тривалість дій і вартість.
9. Сформувати розклад.
10. Створити бюджет.
11. Планувати якість - створити метрики.
12. Розподілити ролі і відповідальності.
13. Створити план комунікацій.
14. Спланувати управління ризиками, ідентифікувати ризики, якісний аналіз, кількісний аналіз, планувати реагування на ризик.

Типові помилки планування.

Планування з використанням помилкових цілей. Будь-який проєкт за своїм змістом призначений для вирішення проблеми, задоволення конкретної потреби і т.ін. Залежно від цього формулюються ті чи інші конкретні цілі. Якщо проблема незрозуміла і недостатньо чітко сформульована, то можна зіткнутися з істотними помилками.

Планування на основі неповних даних. Подібна ситуація характерна для інженерингових проєктів, для яких на результати планування істотно впливають майбутні результати тестування або результати пошукових робіт суміжних напрямів. При цьому доводиться планувати роботи, початок яких, а можливо і сам факт виконання, залежить від результатів тестових випробувань або успіхів/невдач в сусідніх підрозділах.

Планування здійснюється із зауваженням тільки плановиків. Хоча з багатьох причин це виправдано, подібна організація планування може привести до істотних втрат через відсутність обліку важливих факторів. Тому повинні також

залучатися для планування відповідальні виконавці по конкретних робіт проєкту, відповідальні за проєктне фінансування, за поставки і т.ін.

Планування без урахування попереднього досвіду. Навіть при наявності найкращою кошторису без використання попереднього досвіду реалізації аналогічних проєктів можна допустити серйозних помилок в плануванні.

Планування ресурсів без урахування їх доступності. Це стосується перш за все трудових ресурсів, що володіють певною кваліфікацією і можливістю прибути до заданого терміну в задане місце для виконання робіт за проєктом.

Планування без урахування координації. Будь-який, досить великий проєкт розбивається на відносно незалежні частини, за реалізацію яких відповідають самостійні підрозділи. При відсутності координуючих впливів з боку керівника проєкту вони можуть діяти, переслідуючи виключно свої приватні, локальні цілі, що призводить до хаосу і зриву реалізації проєкту в цілому.

Планування без урахування мотивації. Як правило, для робіт по проєктам залучаються виконавці з функціональних підрозділів, у яких є своє керівництво, свої цілі і специфічні завдання і, зрозуміло, своя форма оплати праці, які зазвичай ніяк не пов'язані з цілями і завданнями проєкту. Тому виконавці не відчувають відповідальності і важливості робіт за проєктом без належного стимулювання за результати своєї діяльності. А керівник проєкту не наділений достатніми правами щодо стимулювання виконавців і не може формувати бюджет матеріального стимулювання за результатами в проєкті.

Планування із зайвою деталізацією. Коли проєкт планується занадто детально, виникають проблеми при аналізі, плануванні та контролі його стану (наприклад, що виконано і в чому затримка). Більш того, важко ефективно управляти великою кількістю ресурсів, визначати затримки по часу, оцінювати витрати, розробляти реальні, прийнятні для цілей управління графіки. Проте зайве укрупнення теж може привести до проблем втрати керованості. Необхідна золота середина, коли в проєкті плануються тільки ті параметри, якими можна і потрібно управляти.

Планування не для відстеження. На жаль, це найбільш поширена помилка, коли планування виконується заради того, щоб був план. Всі помилки планування, перераховані в цьому розділі, можуть стати причиною негативного ставлення до плану, коли він перестає бути реальним інструментом управління роботами по проєкту.

Питання до розділу 4

1. Як визначається процес планування проєкту?
2. Що є об'єктами планування в проєкті?
3. Процес розробки планів та параметри реалізації проєкту?
4. Складові плану управління проєктом?
5. Що входить до робіт менеджера з планування проєкту?
6. Які допоміжні роботи при плануванні проєкту?
7. Що таке структурна декомпозиція робіт (СДР)?
8. Етапи розробки та принципи формування структури WBS?
9. Рівні проєкту?
10. Що становить робочий пакет і яку інформацію він повинен відображати?
11. Приклад алгоритму планування ІТ-проєкту?
12. Типові помилки планування проєкту?

РОЗДІЛ 5. ФОРМУВАННЯ ЗМІСТУ ПРОЄКТУ

5.1. Зміст проєкту

Зміст проєкту – це опис робіт, які необхідно виконати, щоб отримати продукт. Для опису всіх необхідних робіт по проєкту потрібно: визначитися з *вимогами і очікуваннями замовника*, визначити, які з них *реально можуть бути здійснені*, і які *ресурси* знадобляться.

За методологією PMI (PMBoK) для цього визначені такі кроки:

1. Зібрати і формалізувати *вимоги до проєкту*.
2. Сформувати *концепцію*.
3. Створити ICP – *ієархічну структуру робіт* (WBS - Work Breakdown Structure).

Зібрати і формалізувати вимоги – один з найбільш *трудомістких і неформалізованих* процесів. Все що відомо напочатку робіт – це *загальні вимоги*, зафіковані в *паспорті проєкту* (цілком можливо, що вони описані кількома реченнями).

Необхідно *конкретизувати вимоги* і необхідно визначити:

- «*хто?*» є джерелом вимог на проєкті;
- *що?* конкретно замовник та користувач хоче та може *отримати* після закінчення робіт.

Практично неможливо на даному етапі призначити відповідальним за всі вимоги *спонсора або замовника*. Хто б не підписав наш паспорт проєкту (устав, статут), а, в подальшому, і контракт на виконання робіт – він *не може бути необхідним джерелом інформації* про вимоги до проєкту.

Кращі відомі проектні практики свідчать (PMBoK):

- проєкт з *незадоволеними очікуваннями замовника* не є успішним;
- проєкт, результати якого *не використовуються* кінцевими користувачами, не є успішним.

За якими ознаками ми визначаємо ту чи іншу людину як зацікавлену особу:

- По-перше, це кожен, хто *прямо залучений в проєкт* (замовник, спонсор, команда проєкту).
- По-друге, зацікавленою особою завжди є *кінцеві користувачі ІТ-продукту*.
- По-третє, *керівництво та члени команди проєкту*.

- По-четверте, ці особи, що безпосередньо *не пов'язані* з виконанням проєктом, але мають вплив на проєкт (фінансові, контролюючі, законодавчі структури та інші).

5.2. Формування вимог

Очікування – це абстрактна картинка майбутнього проєкту.

Приклад очікування: «щоб продуктивність відділу зросла після впровадження ІТ-системи»; або «щоб впровадження проєкту не сильно позначилося на роботі сусіднього департаменту».

Очікування не можна включити до складу проєкту, не перетворивши у вимоги.

Вимога – конкретний, вимірний *параметр* ІТ-проєкту, що відповідає побажанням зацікавленої особи.

Приклад вимоги: «ІТ-система повинна дозволяти виконувати всі призначенні для користувача сценарії за прототипами».

Належить вибрати один або кілька методів отримання від зацікавлених осіб їх очікувань і вимог.

Головним чином тут важлива інформація *від представників замовника*.

5.3. Методи збору інформації щодо очікувань та вимог замовника

Вибираючи методи, дуже важливо ретельно визначити та оцінити потреби в інформації щодо очікувань та вимог і здатності представників замовника надати достовірну та повну інформацію.

З найбільш поширеніх можна виділити:

- *Інтерв'ю*.
- *Анкетування (Опитувальники)*.
- *Мозкові штурми* (в різних варіантах та комбінаціях).
- *Прототипування*.

Інтерв'ю – є одним з найбільш надійних методів, він же – самий трудомісткий.

Головний плюс – безпосереднє спілкування дозволяє зібрати *найбільшу повну і достовірну інформацію*, а також встановити конструктивний робочий контакт зі співрозмовником.

Головний мінус – *трудовитрати* (доведеться витрачати час багатьох учасників у великих кількостях).

Складності:

– Перша причина – з зацікавленими особами спілкування може виявитися неможливим через їх *посадовий статус*. Не завжди можливо умовити топ-менеджмент замовника приділити вам годинку-другу на очній зустрічі, навіть якщо це в інтересах проєкту.

– Друга причина – часто важко визначити представників замовника, що дійсно *розуміють вимоги та очікування* від ІТ-проєкту.

Анкетування (Опитувальники) – це ефективний та достатньо точний спосіб швидко зібрати інформацію від великої кількості представників замовника та інших зацікавлених осіб (таблиця 5.1).

При цьому є можливість надавати (вводити та надсиляти) інформацію у зручний час для представників замовника та інших зацікавлених осіб.

Переваги – можливість отримання поглядів на ці питання *від багатьох осіб* для подальшого хі оцінювання.

Недоліки – висока ймовірність формального підходу до заповнення анкет (за принципом «щоб відстали»).

Таблиця 5.1 Приклад анкети процесу преміювання

Приклад анкети

Посада/ роль/ ПІБ	Отримує документ/ наказ/ сповіщення	Від кого отримує	Як реагує/ оброблює/ які дії виконує	Передає документ/ наказ/ сповіщення	Кому передає
...					
Директор	-	-	Готує наказна преміювання	Наказ на преміювання. раз на місяць	Бухгалтеру
Бухгалтер	Наказ на преміювання. раз на місяць	Директора	Готує відомості	Відомості на премію	Директору на затвердження
Директор	Відомості на премію	Бухгалтера	Затверджує	Затверджені відомості на премію	Касиру
Касир	Затверджені відомості на премію	Директора	Проводить видачу премій під підпис працівникам	Відомості з підписами	Бухгалтеру

Мозковий штурм (*brainstorming*) популярний метод висування творчих ідей у процесі розв'язування наукової чи технічної проблеми, сеанси якого стимулюють творче мислення.

Мозковий штурм – можна назвати «колективним інтерв'ю».

Проведений за певними правилами, мозковий штурм може виявитися вкрай ефективним при виявленні вимог та очікувань при формуванні змісту ІТ-проекту.

Під час мозкового штурму є можливість *перевірити та усереднити результати попередніх кроків*.

Однак деяким людям важко спілкуватися навіть в невеликих колективах і вони можуть «відмовчуватися» і соромилися у висловлюваннях не маючи навички участі у подібних заходах.

Прототипування – це дуже гарний спосіб зібрати або уточнити вимоги.

Під прототипом ми можемо розуміти будь-який зrozумілий вашому співрозмовнику образ продукту (будь-то блок-схема, модель бізнес-процесу або приклад аналогічної іншої ІТ-системи).

Існують спеціальні програмні засоби побудови прототипів.

Прототипування ефективно поєднувати з іншими техніками (наприклад, інтерв'ю, анкетуванням); головне не обмежуватися тільки ним, щоб *не упустити суттєві моменти, не реалізовані в прототипі*, але *важливі для продукту проєкту*, що розроблюється.

Збір вимог в рамках початкового планування повинен бути *розумно-глибоким і максимально широким*. При цьому мова *не йде* про те, що на цьому *робота з вимогами повинна припинитися*, а їх список буде «*фіксований*» до *кінця проєкту*.

Робота з очікуваннями замовника, накопичення його вимог, коригування планів протягом всього життєвого циклу проєкту – це є основою проєктного управління.

Іноді процес первісного збору вимог виявляється стільки *трудомістким і розтягнутим в часі*, що доцільно виділити його *в окремий проєкт*.

5.4. Балансування вимог

Балансування вимог – це відбір вимог, *відображення та реалізація* яких передбачається в рамках проєкту.

Процес балансування заснований на поєднанні *досвіду та інтуїції виконавця (менеджера проєкту)* та можливостей реалізації у рамках *можливого бюджету* проєкту.

Спершу визначаються *приоритети вимог* (виділяючи найбільш значущі), а потім відбираємо до реалізації ті з них, які можливо укласти в *рамки проектних обмежень (вартості, часу, інших ресурсів та реальних можливостей)*.

Варто пам'ятати, що деякі вимоги можуть виявитися такими що *взаємопов'язані і виключають одна одну*. Тобі необхідно визначитись з *конфліктом вимог*, як елементом балансування вимог.

На даному етапі ми маємо лише *попередні досить неточні оцінки структури, вартості та термінів проєкту* і поки не знаємо точно, скільки коштуватимуть і будуть тривати обрані для виконання роботи. У поточних оцінках менеджер проєкту покладається на професійний досвід та інтуїцію як свою, так і команди проєкту.

Пізніше, коли *собівартість і тривалість робіт буде остаточно оцінена*, можна повернутися до цієї частини плану і скорегувати її. Можливо, *від якихось вимог доведеться відмовитися або якісь додати чи переглянути*.

Однак важливо розуміти – *процес балансування вимог не повинен суперечити паспорту (статуту, уставу) проєкту*.

5.5. Концепція проєкту (scope)

Після збору і балансуванню вимог треба визначитися – *які дії по проєкту необхідні*, щоб виконати задумане. Ми вже знаємо, що очікує та вимагає замовник та майбутній користувач IT-проєкту.

Концепція проєкту **важлива** для команди розробників, але не для усіх інших учасників проєкту.

Результатом опису та формалізації вимог є концепція проєкту. Даний документ буде містити як *загальну інформацію про проєкт*, так і посилання на всілякі *вимоги і опис продукту*, так що кожний учасник проєкту, в залежності від характеру своїх питань зможе самостійно знайти максимум інформації без сторонньої допомоги.

Не менш важливо і те, що «концепція» містить опис «проектного підходу», бо вона поєднує усі елементи та особливості проєкту і при цьому чітко *визначає межі проєкту*.

Треба зробити, щоб scope був доступним і проінформувати про нього всіх учасників.

Но основі концепції проєкту готується технічне завдання до контракту виконання проєкту.

Структурна декомпозиція робіт (*СДР*) проєкту – це *одна з головних частин змісту проєкту*, яка визначає *розділення проєкту на складові частини* (завдання,

роботи, задачі), з деталізацією, яка необхідна і достатня для його ефективного планування, моніторингу та управління.

До основних задач розробки СДР відносяться:

- визначення **ступеня деталізації** проектних робіт (так, щоб вони піддавались оцінці);
- визначення **кількості рівнів** (як правило три-чотири, для сучасних компаній - чотири оптимально);
- розробка **структурі кожного рівня** (формуються горизонтальні рівні);
- підготовка **опису елементів** СДР (стисла назва кожної складової СДР);
- формування **системи кодування** (кодуються всі блоки);
- проведення **обчислень витрат** (розраховуються витрати по кожному блоку).

Можна визначити такі основні типові помилки планування змісту проєкту:

Планування з використанням помилкових цілей. Будь-який проєкт за своїм змістом призначений для вирішення проблеми, задоволення конкретної потреби і т.ін. Залежно від цього сформулюються ті чи інші конкретні цілі. Якщо проблема незрозуміла і недостатньо чітко сформульована, то можна зіткнутися з істотними помилками.

Планування на основі неповних даних. Подібна ситуація характерна для інженерних проєктів, для яких на результати планування істотно впливають майбутні результати тестування або результати пошукових робіт суміжних напрямів. При цьому доводиться планувати роботи, початок яких, а можливо і сам факт виконання, залежить від результатів тестових випробувань або успіхів/невдач в сусідніх підрозділах.

Планування здійснюється із залученням тільки плановиків.Хоча з багатьох причин це вправдано, подібна організація планування може привести до істотних втрат через відсутність обліку важливих факторів. Тому повинні також залучатися для планування відповідальні виконавці конкретних робіт проєкту, відповідальні за проєктне фінансування, за поставки і т.ін.

Планування без урахування попереднього досвіду. Навіть при наявності найкращою кошторису без використання попереднього досвіду реалізації аналогічних проєктів можна допустити серйозних помилок в плануванні.

Планування ресурсів без урахування їх доступності. Це стосується перш за все трудових ресурсів, що володіють певною кваліфікацією і можливістю прибути до заданого терміну в заданий місце для виконання робіт за проєктом.

Планування без урахування координації. Будь-який, досить великий проект розбивається на відносно незалежні частини, за реалізацію яких відповідають самостійні підрозділи. При відсутності координуючих впливів з боку керівника проекту вони можуть діяти, переслідуючи виключно свої приватні, локальні цілі, що призводить до хаосу і зриву реалізації проекту в цілому.

Планування без урахування мотивації. Як правило, для робіт по проектам залучаються виконавці з функціональних підрозділів, у яких є своє керівництво, свої цілі і специфічні завдання і, зрозуміло, своя форма оплати праці, які зазвичай ніяк не пов'язані з цілями і завданнями проекту. Тому виконавці не відчувають відповідальності і важливості робіт за проектом без належного стимулювання за результати своєї діяльності. А керівник проекту не наділений достатніми правами щодо стимулювання виконавців і не може формувати бюджет матеріального стимулювання за результатами в проекті.

Планування із зайвою деталізацією. Коли проект планується занадто детально, виникають проблеми при аналізі, плануванні та контролі його стану (наприклад, що виконано і в чому затримка). Більш того, важко ефективно управляти великою кількістю ресурсів, визначати затримки по часу, оцінювати витрати, розробляти реальні, прийнятні для цілей управління графіки. Проте зайве укрупнення теж може привести до проблем втрати керованості. Необхідна золота середина, коли в проекті плануються тільки ті параметри, якими можна і потрібно управляти.

Планування не для відстеження. На жаль, це найбільш поширена помилка, коли планування виконується заради того, щоб був план. Всі помилки планування, перераховані в цьому розділі, можуть стати причиною негативного ставлення до плану, коли він перестає бути реальним інструментом управління роботами по проекту.

Питання до розділу 5

1. Зміст проекту?
2. Збір вимог, виявлення компетентних осіб та формування вимог?
3. Методи збору інформації щодо очікувань та вимог замовника?
4. Що таке балансування вимог?
5. Концепція проекту?
6. Структурна декомпозиція робіт. Її задачі?
7. Типові помилки планування змісту проекту?

РОЗДІЛ 6. ВИКОРИСТАННЯ ДІАГРАМИ ГАНТА ДЛЯ ПЛАНУВАННЯ РОБІТ ПРОЄКТУ

Генрі Лоуренс Гант (20 травня 1861 - 23 листопада 1919 р.) був американським інженером-механіком і консультантом з менеджменту, найбільш відомим своєю роботою в галузі розвитку наукового менеджменту. Він створив діаграму Ганта у 1910-х роках (рис. 6.1).

Діаграми Ганта використовувалися у великих інфраструктурних проєктах, включаючи греблю Гувера та систему автомагістралей між штатами, і продовжують залишатися важливим інструментом в управлінні проєктами та програмами.

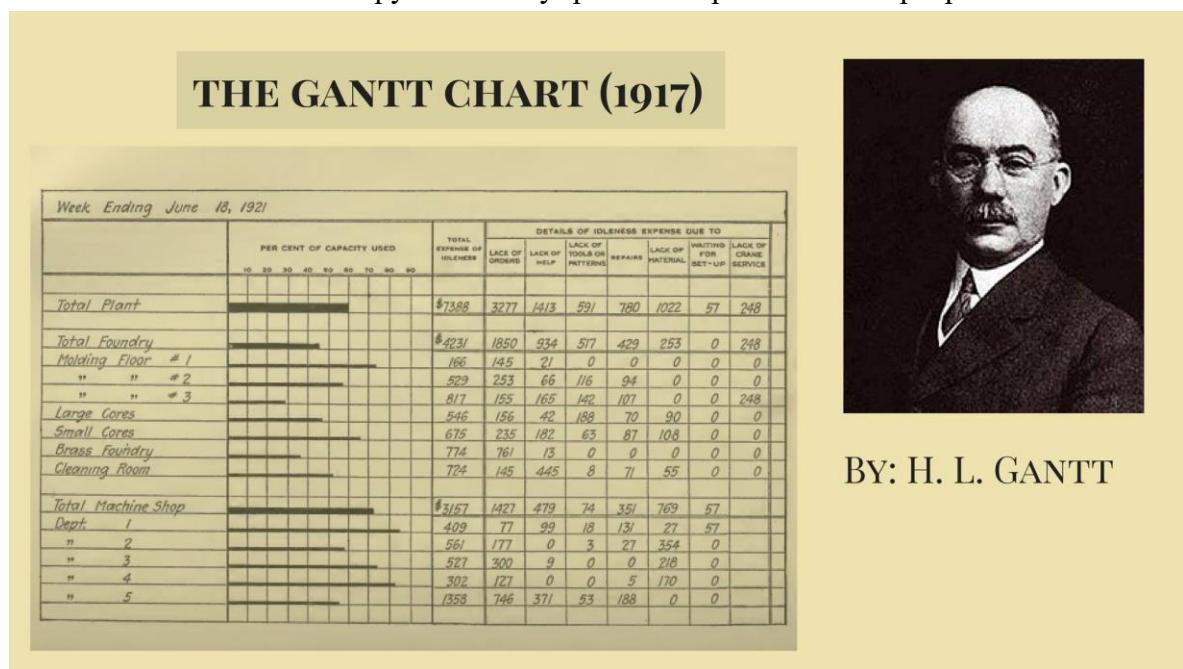


Рис. 6.1 Діаграма Ганта

6.1. Призначення діаграми Ганта

Діаграма Ганта – це методологія представлення діяльності або завдань, яка має на меті дати узагальнене уявлення про час, витрачений на кожну задачу, що розглядається незалежно в процесі.

Іншими словами, діаграма Ганта – це тип діаграми, що представляє діяльність самостійно, щоб мати загальну картину того, як завдання еволюціонують з часом.

Діаграма дозволяє:

- відстежувати прогрес та взаємодію між задачами;
- делегувати задачі та декомпозувати;
- ефективно розподіляти ресурси;
- визначати та дотримуватись термінів виконання;
- призначати відповідальних за завдання осіб;
- визначати та контролювати вартість задач та проєкту у цілому.

Діаграма Ганта (Gantt chart, або стрічкова діаграма, графік Ганта) – це тип стовпчастих діаграм, який використовується для ілюстрації плану-графіка робіт проєкту.

Діаграма Ганта є одним із методів планування проєктів.

Діаграма Ганта використовується як інструмент для проєктування поставлених завдань і управління ними.

На діаграмі Ганта зображуються:

- конкретні завдання (задачі) проєкту;
- їх послідовність;
- зв'язки між ними, що дає можливість виконувати всі в призначений термін.

По вертикалі – відображаються відрізки, які являють собою окремий проєкт або завдання (задачі).

По горизонталі – часова шкала, що дозволяє відображати та контролювати час виконання та часові обмеження для кожної з задач і проєкту у цілому. Тобто, кожна задача має свої часові обмеження.

6.2. Склад діаграми Ганта

Діаграма Ганта складається із відрізків, які розміщені на горизонтальній шкалі часу.

Кожен відрізок представляє собою певне завдання чи задача.

Початок, кінець і довжина відрізку відповідає *початку, завершенню та тривалості* завдання.

Завдання можуть виконуватися як *паралельно так і послідовно*. Якщо завдання виконуються послідовно, то існує зв'язок між ним і попередньою задачею відповідно.

Наступна задача буде виконуватися *тільки після завершення попередньої*.

Паралельні завдання в проєкті потрібно починати якнайшвидше, що дає змогу зекономити час і тривалість виконання проєкту.

Заштрихована область в стрічці показує *процент виконання* конкретного завдання. Таким чином виконується контроль.

В діаграмі Ганта часто використовують таблиці і надписи, які більш детально описують завдання, залученість матеріальних та людських ресурсів у проєкті.

6.3. Побудова діаграми Ганта

Діаграма Ганта будується на основі *термінів створення продукту проєкту і кошторису* на виконання робіт проєкту.

На цій основі створюється таблиця для проєкту, в яку вносяться такі основні дані:

- **назва завдання (задачі);**
- **дата початку;**
- **дата закінчення;**
- **вартість робіт** (оплата за виконання);
- **виконавець** (відповідальний).

6.4. Приклад побудови діаграми Ганту

Попередньо готуються переліки задач у Excel чи Word (рис. 6.2).

	A	
1	task 1	
2	task 2	
3	task 2.1	
4	task 2.2	
5	task 3	
6	task 3.1	
7	task 3.1.1	
8	task3.1.2	
9	task 3.1.3	
10	task 3.2	
11	task 4	
12	task 5	

task 1	
task 2	
task 2.1	
task 2.2	
task 3	
task 3.1	
task 3.1.1	
task3.1.2	
task 3.1.3	
task 3.2	
task 4	
task 5	

Рис. 6.2 Підготовка завдання у Excel чи Word

Крок 1. Створюємо в Project Libre «Новий проект» (рис. 6.3).

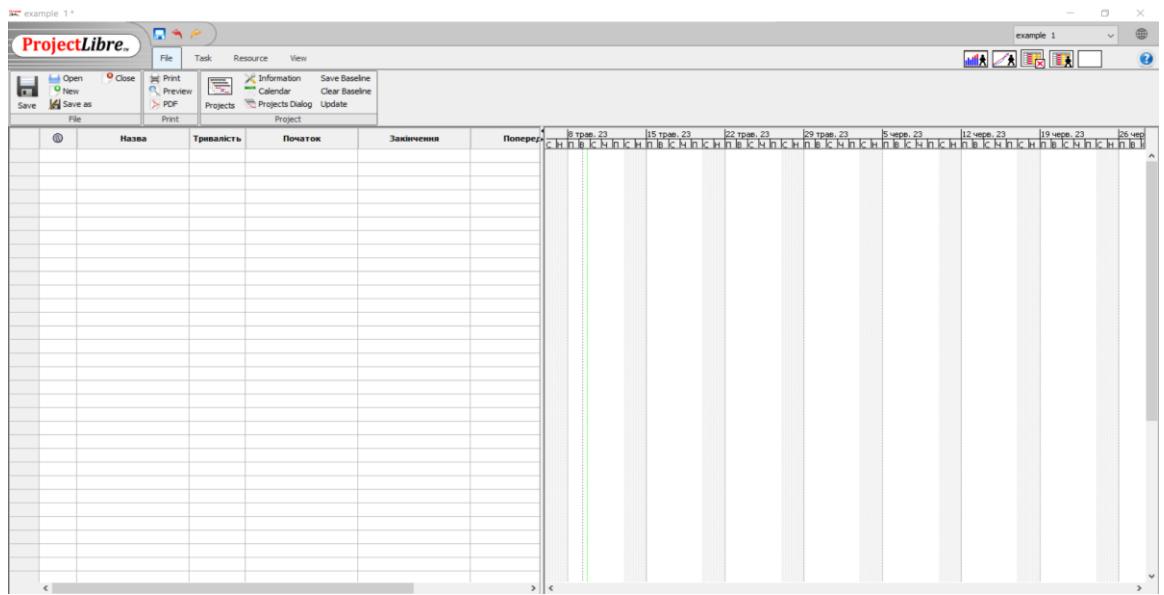


Рис. 6.3 Створення в Project Libre «Новий проект»

Крок 2. Скопіюємо список до проєкту (рис. 6.4).

	Назва	Тривалість	Початок	Закінчення	Попередник
1	task 1	1 день?	09.05.23 8:00	09.05.23 17:00	
2	task 2	1 день?	09.05.23 8:00	09.05.23 17:00	
3	task 2.1	1 день?	09.05.23 8:00	09.05.23 17:00	
4	task 2.2	1 день?	09.05.23 8:00	09.05.23 17:00	
5	task 3	1 день?	09.05.23 8:00	09.05.23 17:00	
6	task 3.1	1 день?	09.05.23 8:00	09.05.23 17:00	
7	task 3.1.1	1 день?	09.05.23 8:00	09.05.23 17:00	
8	task3.1.2	1 день?	09.05.23 8:00	09.05.23 17:00	
9	task 3.1.3	1 день?	09.05.23 8:00	09.05.23 17:00	
10	task 3.2	1 день?	09.05.23 8:00	09.05.23 17:00	
11	task 4	1 день?	09.05.23 8:00	09.05.23 17:00	
12	task 5	1 день?	09.05.23 8:00	09.05.23 17:00	

Рис. 6.4 Копіювання списку до проєкту

Крок 3. Вставимо сумарну задачу (рис. 6.5).

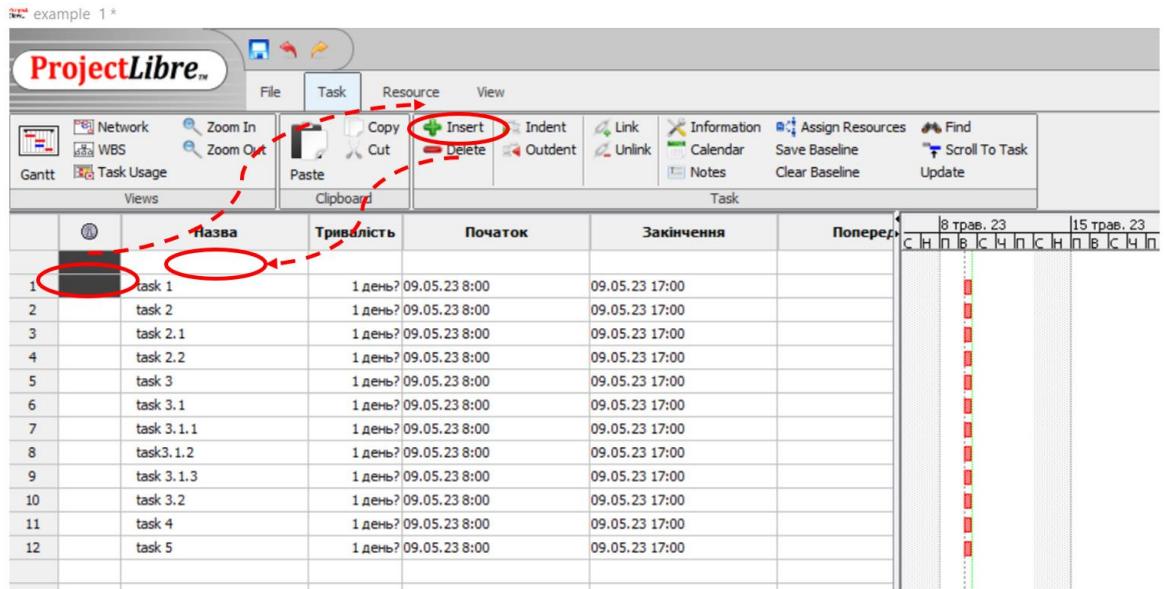


Рис. 6.5 Сумарна задача

Виділимо першу задачу, натиснемо 'Insert' та створимо сумарну задачу.

Крок 4. Встановимо ієрархію декомпозиції задач шляхом здигу направо відповідних задач (команди 'Indent' та 'Outdent') (рис. 6.6).

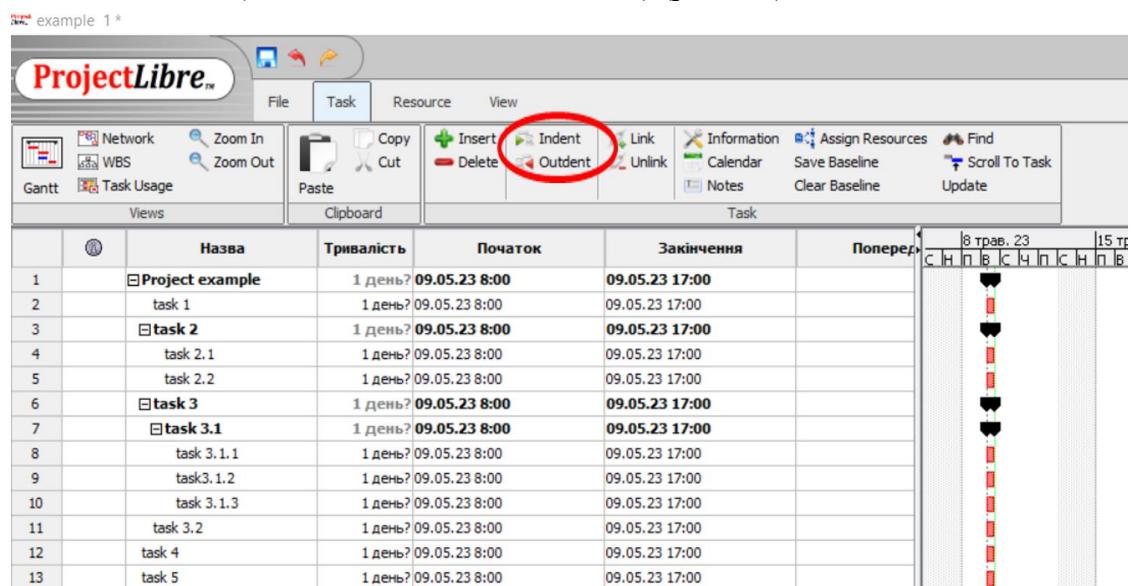


Рис. 6.6 Ієрархія декомпозиції задач

Крок 5. Визначимо терміни виконання робіт (рис. 6.7).

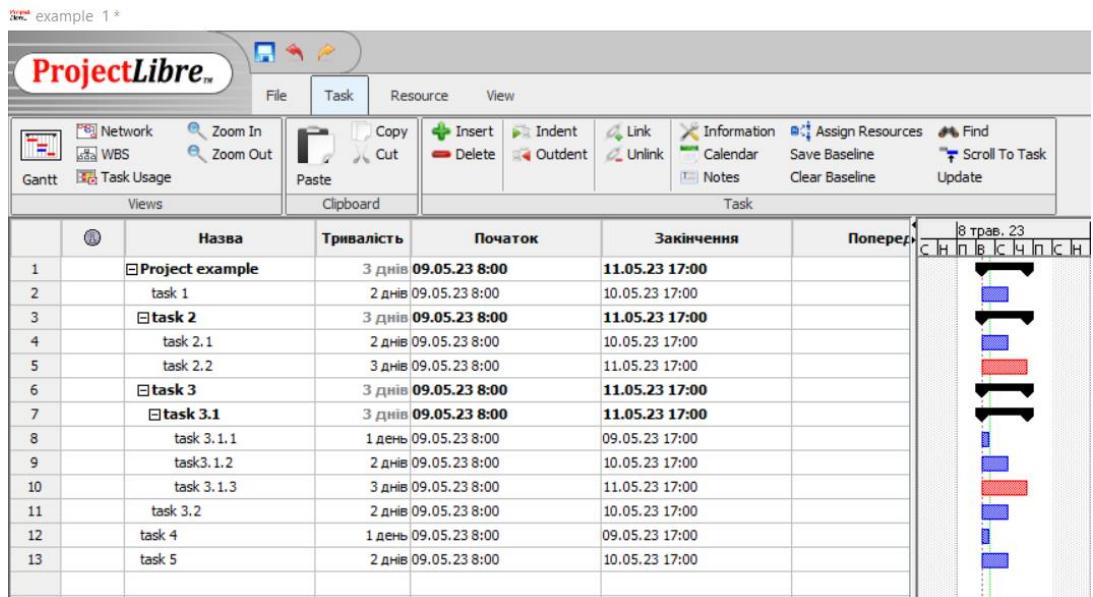


Рис. 6.7 Терміни виконання робіт

Крок 6. Встановимо зв’язки між задачами відповідно до їх ієархії (попередники-наслідники) цифровим чи графічним методом (рис. 6.8).

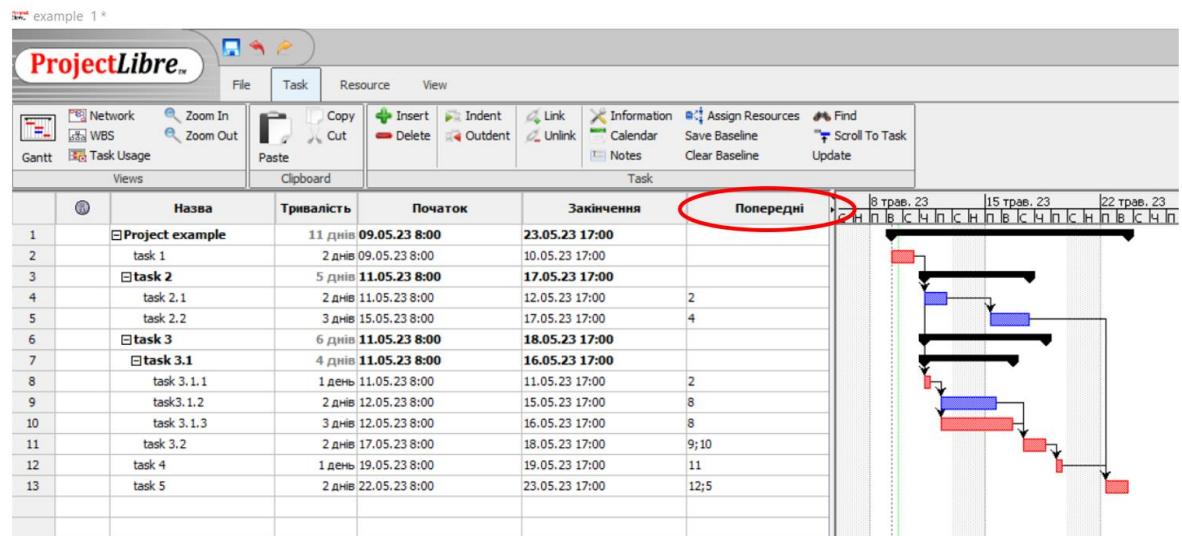


Рис. 6.8 Зв’язки між задачами

Крок 7. Визначаємо перелік ресурсів – виконавців та їх тарифи оплати (рис. 6.9).

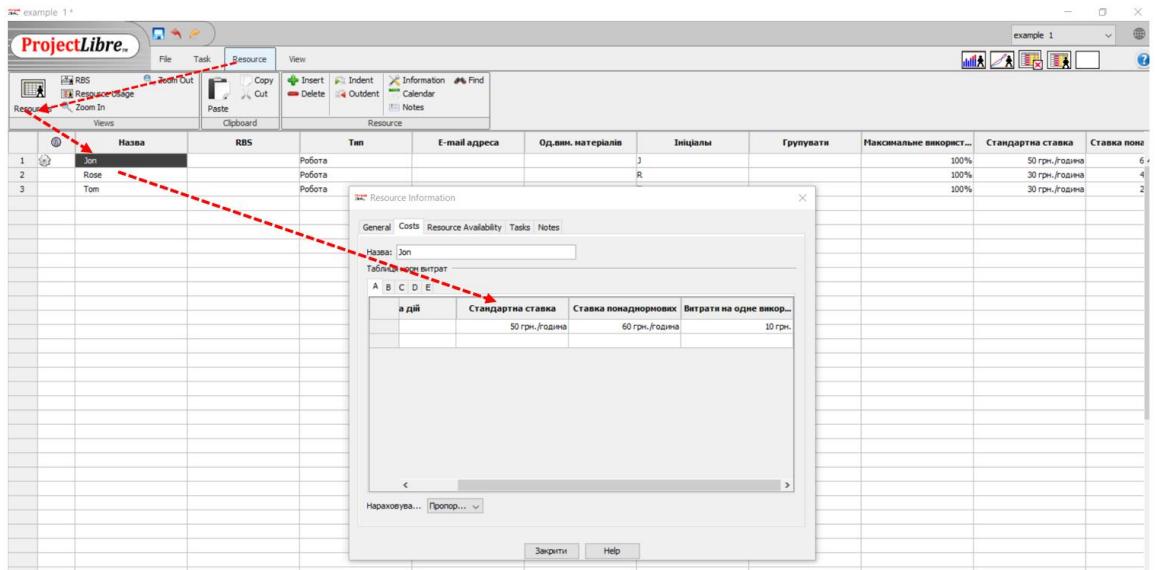


Рис. 6.9 Перелік ресурсів – виконавців та їх тарифи оплати

Крок 8. Призначення ресурсів (рис. 6.10).

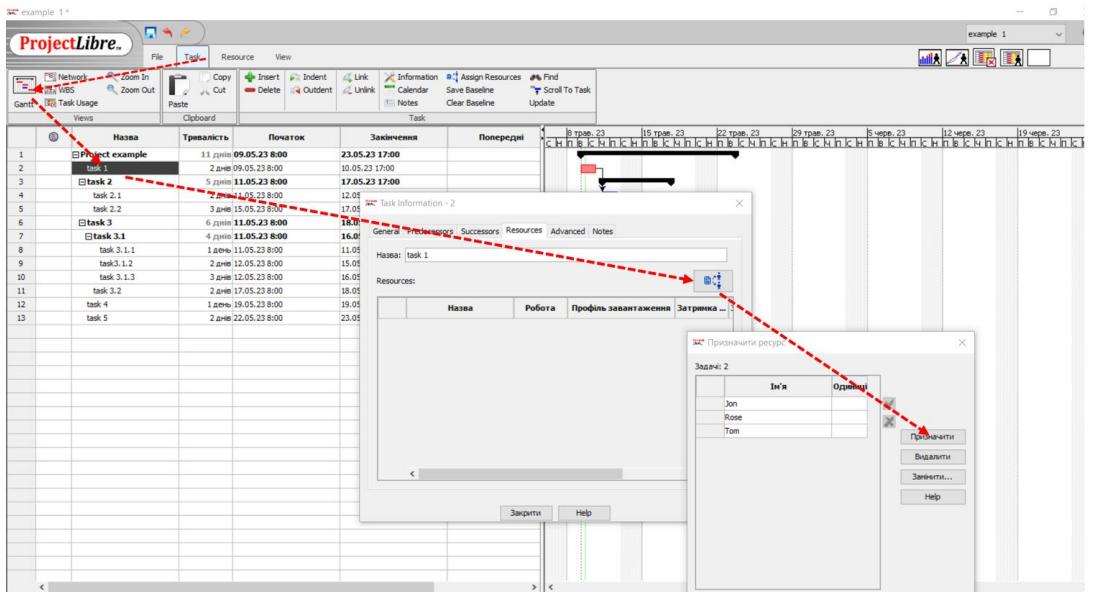


Рис. 6.10 Обрання функції «Призначення ресурсів»

Всі виконавці призначені (рис. 6.11).

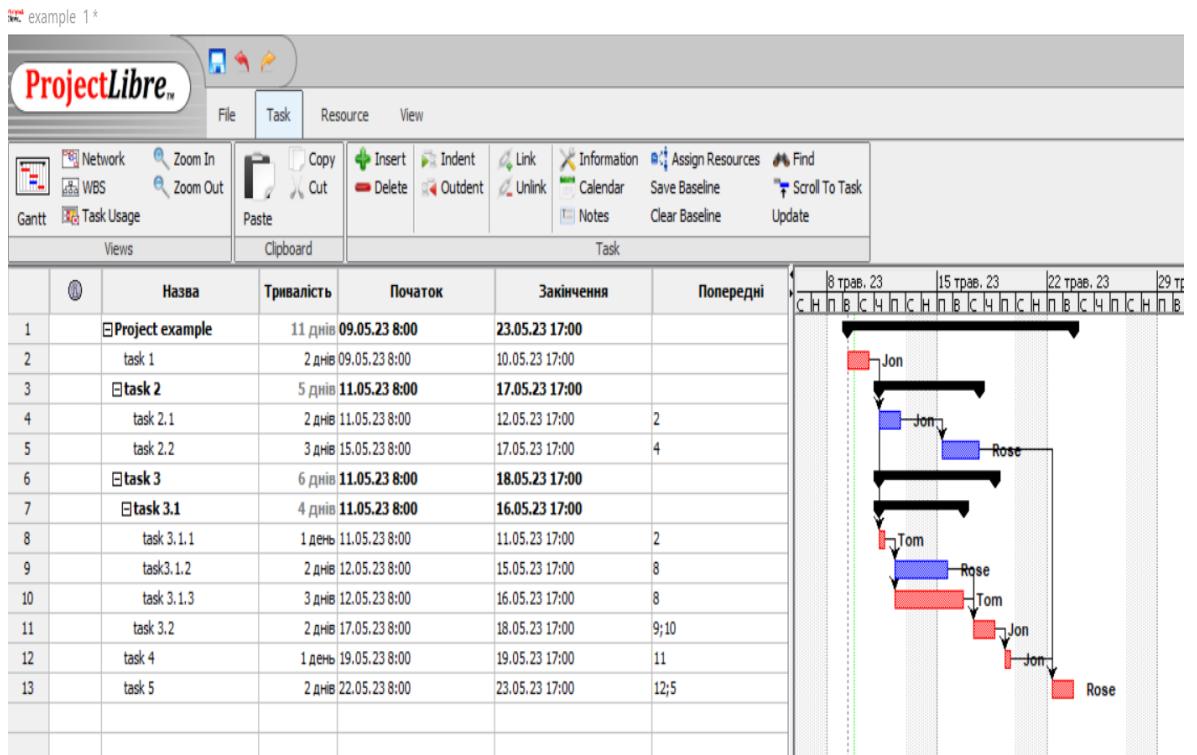


Рис. 6.11 Призначені виконавці

Крок 9. Додавання колонок (рис. 6.12).

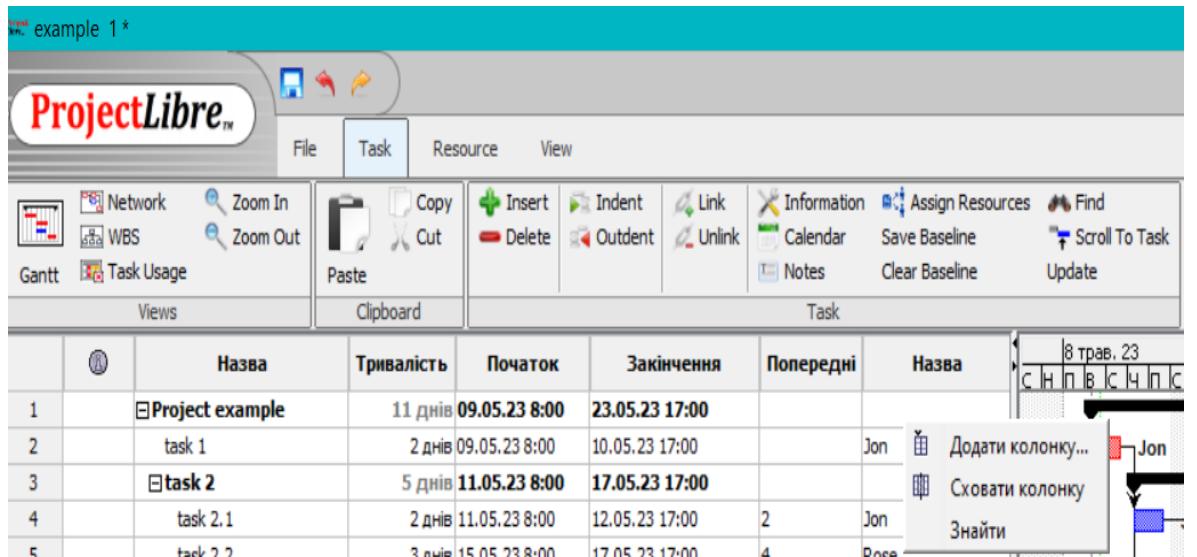


Рис. 6.12 Нова колонка

Крок 9. Додаткові колонки ‘Робота’ та ‘Вартість’ (рис. 6.13).

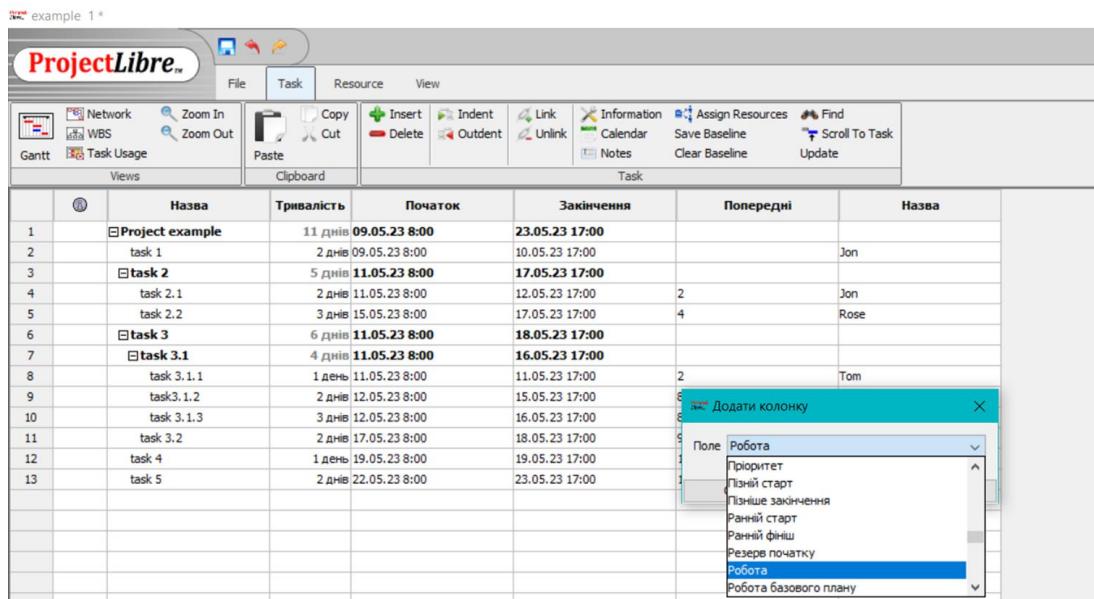


Рис. 6.13 Додаткові колонки «Робота» та «Вартість»

Проект сформовано (рис. 6.14).

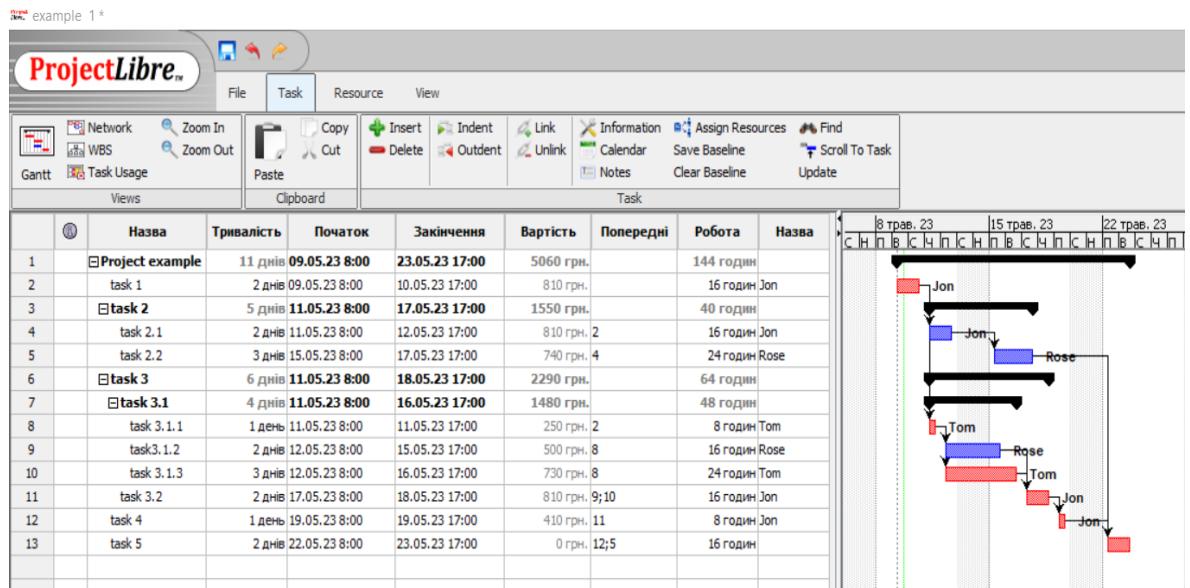


Рис. 6.14 Загальний вигляд сформованого проекту

Критичний шлях – це максимальний за тривалістю повний шлях у проекті від початку і до завершення реалізації проекту.

Роботи, що лежать на цьому шляху, також називаються *критичними*.

Саме тривалість критичного шляху визначає найменшу можливу загальну тривалість робіт проєкту в цілому.

Для успіху проєкту с точки зору вірогідності його своєчасного, якісного та без перевищення бюджету, виконання бажано щоб на критичному шляху було б не більш 70% задач.

На діаграмі Ганта у проєкті відображається загальний часовий резерв робіт та червоним кольором критичний шлях (рис. 6.15).

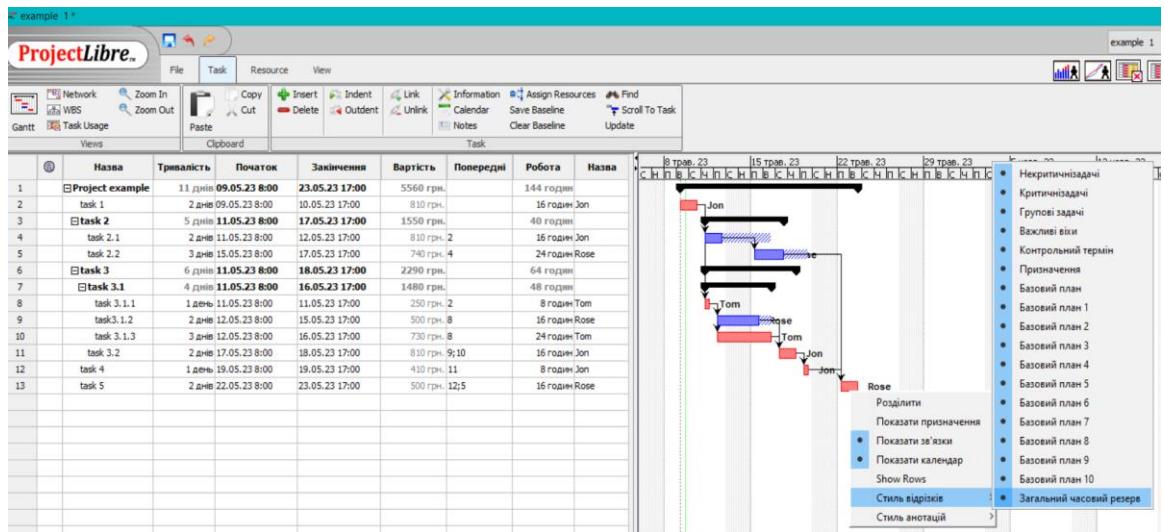


Рис. 6.15 Загальний вигляд

Особливості відображення діаграми Ганта:

- *усі задачі у діаграмі мають вхід та вихід* (крім першої – тільки вихід та останньої – вхід);
- *одна задача* може мати *кілька входів та виходів*;
- може бути *кілька перших та останніх задач* у діаграмі.

Питання до розділу 6

1. Що таке діаграма Ганта. Її призначення та можливості?
2. Склад та побудова діаграми Ганта?
3. Кроки побудови діаграми Ганта?
4. Що таке критичний шлях?
5. Критичний шлях на діаграмі Ганта?

РОЗДІЛ 7. ЗВ'ЯЗКИ ТА КОМУНІКАЦІЙ ПРОЄКТУ

7.1. Управління комунікаціями проєкту (Project Communication Management)

Управління комунікаціями проєкту є задачею управління взаємодією інформаційних зв'язків і, як управлінська функція, ця задача спрямована на забезпечення:

- своєчасного збору інформації *про стан проєкту*;
- генерації звітів виконавців та звернень до учасників проєкту;
- розподілу даних про *потреби та ризики проєкту*;
- збереження необхідної *проєктної інформації*.

Під інформацією розуміють *зібрані, оброблені і розподілені дані про процес та результати роботи за проєктом*.

Для прийняття рішень важливо, щоб інформація повинна бути надана:

- *своєчасно*;
- *за призначенням*;
- *в зручній формі*.

Це досягається використанням *сучасних інформаційних технологій* в рамках системи управління проєктом.

Комунікації проєкту і супутня їм інформація є свого роду фундаментом **для забезпечення координації дій** учасників проєкту.

7.2. Основні питання в управлінні комунікаціями



Рис. 7.1 Питання до проєкту

Управління комунікаціями забезпечує (рис. 7.1):

- підтримку системи зв'язку (*взаємодій*) між учасниками проєкту;
- передачу управлінської та звітної інформації, спрямованої на забезпечення досягнення цілей проєкту.

Кожен учасник проєкту повинен *бути підготовлений до взаємодій* в рамках проєкту відповідно до *його функціональних обов'язків*.

Функція управління інформаційними зв'язками включає в себе **наступні процеси**:

- *планування системи комунікацій* – як *визначення інформаційних потреб* учасників проєкту (*склад інформації, терміни і способи доставки*);
- *збір і розподіл інформації* – процеси регулярного *збору і своєчасної доставки необхідної інформації* учасникам проєкту;
- *звітність про хід виконання проєкту – обробка фактичних результатів стану робіт проєкту*, співвідношення з плановими і *аналіз тенденцій, прогнозування*;
- *документування ходу робіт – збір, обробка та зберігання документації по проєкту*.

7.3. Планування системи комунікацій

План комунікацій є складовою частиною плану проєкту включає в себе:

- план *збору інформації*, в якому визначаються джерела інформації і методи її одержання;
- план *розподілу інформації*, в якому визначаються споживачі інформації та способи її доставки;
- детальний *опис кожного документа*, який повинен бути отриманий або переданий, включаючи формат, зміст, рівень детальності і використовувані визначення;
- план *введення в дію* тих чи інших *видів комунікацій*;
- методи *новлення і вдосконалення* плану комунікацій.

План комунікацій формалізується і деталізується в залежності від потреб проєкту.

В рамках проєкту існує *потреба в здійсненні різних видів комунікацій*:

- *внутрішні* (всередині команди проєкту) і *зовнішні* (з керівництвом компанії, замовником, зовнішніми організаціями і т.ін.);

- *формальні* (звіти, запити, наради) і *неформальні* (нагадування, обговорення);
 - *електронні, письмові та усні;*
 - *вертикальні і горизонтальні.*

Системи збору і розподілу інформації повинні забезпечувати потреби різних видів комунікацій. Для цих цілей можуть використовуватися *автоматизовані і неавтоматизовані методи* збору, обробки і передачі інформації.

Неавтоматизовані методи включають збір і *передачу даних на паперових носіях, проведення нарад, телефонні спілкування.*

Автоматизовані методи передбачають використання комп'ютерних технологій і сучасних засобів зв'язку для підвищення ефективності взаємодії: *онлайн конференції, електронна пошта, системи документообігу та архівування даних.*

7.4. Звітність про хід виконання проєкту

Процеси *збору і обробки* даних про *фактичні результати виконання проєкту* та відображення інформації про стан робіт у звітах забезпечують основу *для координації робіт, оперативного планування і управління.*

Звітність про хід виконання включає:

- інформацію про *поточний стан проєкту* в цілому і в розрізі окремих показників;
- інформацію про *відхилення від базових планів;*
- інформацію про *можливі ризики* виконання проєкту;
- *прогнозування* майбутнього стану проєкту.

Основні та проміжні результати ходу робіт повинні бути формально *задокументовані.*

Документування результатів ходу робіт включає в себе:

- *збір і верифікацію* остаточних даних;
- *аналіз і висновки* про ступінь *досягнення результатів* проєкту та ефективності виконаних робіт;
- *архівування результатів* з метою подальшого використання.

Комп'ютерні системи ведення електронних архівів *дозволяють автоматизувати процеси* зберігання і індексації текстових і графічних документів, значно полегшити доступ до архівної інформації.

7.5. Інформаційна система управління проєктом

Інформаційна система управління проєктом – це організаційно-технологічний комплекс методичних, технічних, програмних і інформаційних засобів, спрямований на підтримку і підвищення ефективності процесів управління проєктом.

В процесі реалізації проєкту менеджерам доводиться оперувати значними обсягами даних, які можуть бути зібрані і організовані з використанням комп'ютера.

Розвиток систем управління проєктами для персональних комп'ютерів пройшло через кілька етапів. З збільшенням потужності ПК поліпшувалася функціональність систем, підвищувалися їх можливості. З введенням стандартів обміну даними між системами, поширенням Web-технологій відкрилися нові можливості для подальшого розвитку систем підтримки процесів управління проєктами та їх більш ефективного використання.

Самі проєкти стають все більш складними, що висуває додаткові вимоги до розвитку інформаційних технологій управління проєктами (рис. 7.2).

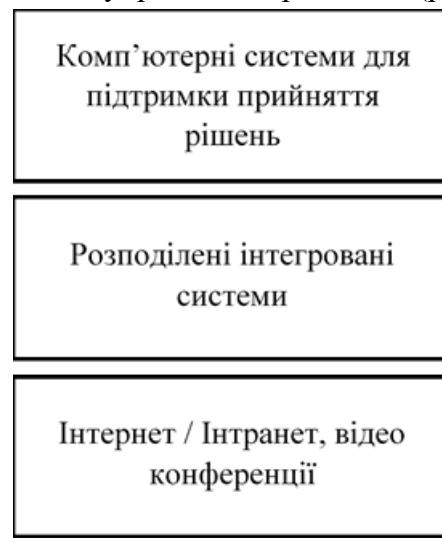


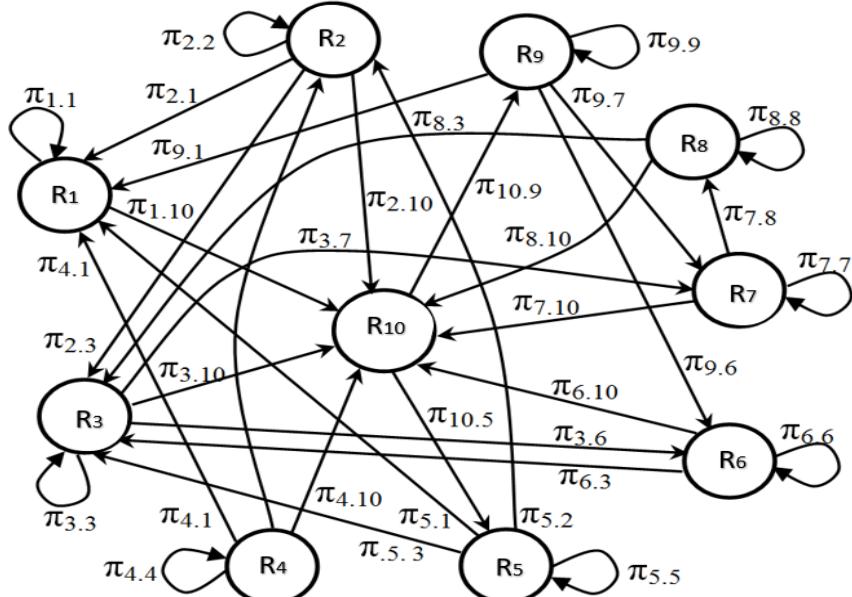
Рис. 7.2 Автоматизація управління проєктом

Програмне забезпечення підтримки групової роботи дозволяє:

- обмін *електронною поштою*;
- *документообіг*;
- *групове планування діяльності*;
- участь віддалених членів команди в *інтерактивних дискусіях* засобами підтримки і *проведення обговорень*;

- проведення "мозкового штурму", який дає можливість його учасникам висловити свою думку за допомогою комп'ютерів, підключених до одного великого екрану.

Зв'язки ролей у проекті – основа комунікації. Для опису інформаційних зв'язків у проекті можуть бути використані граф комунікації ролей (рис. 7.3) у проекті та таблиця комунікацій (таблиця 7.1), які досить повно та багатосторонньо описують можливі інформаційні взаємодії.



$$\text{Максимальна кількість зв'язків } k = n*(n-1)/2$$

Рис. 7.3 Граф, комунікації ролей в проекті

Таблиця 7.1 Таблиця комунікацій до графу комунікацій

Хто надає		Кому надає інформацію								
		R1		R2		...		Rn		
		Що	Коли	Що	Коли	Що	Коли	Що	Коли	
Хто надає	R1	$\pi_{11.}$ Опис		$\pi_{12.}$ Опис				$\pi_{1n.}$ Опис		
	R2	$\pi_{21.}$ Опис								
	...									
	Rn	$\pi_{n1.}$ Опис		$\pi_{n2.}$ Опис				$\pi_{nn.}$ Опис		

Джерела зв'язків у комунікаціях проекту можуть бути також ієрархічна структура задач проекту (рис. 7.4).

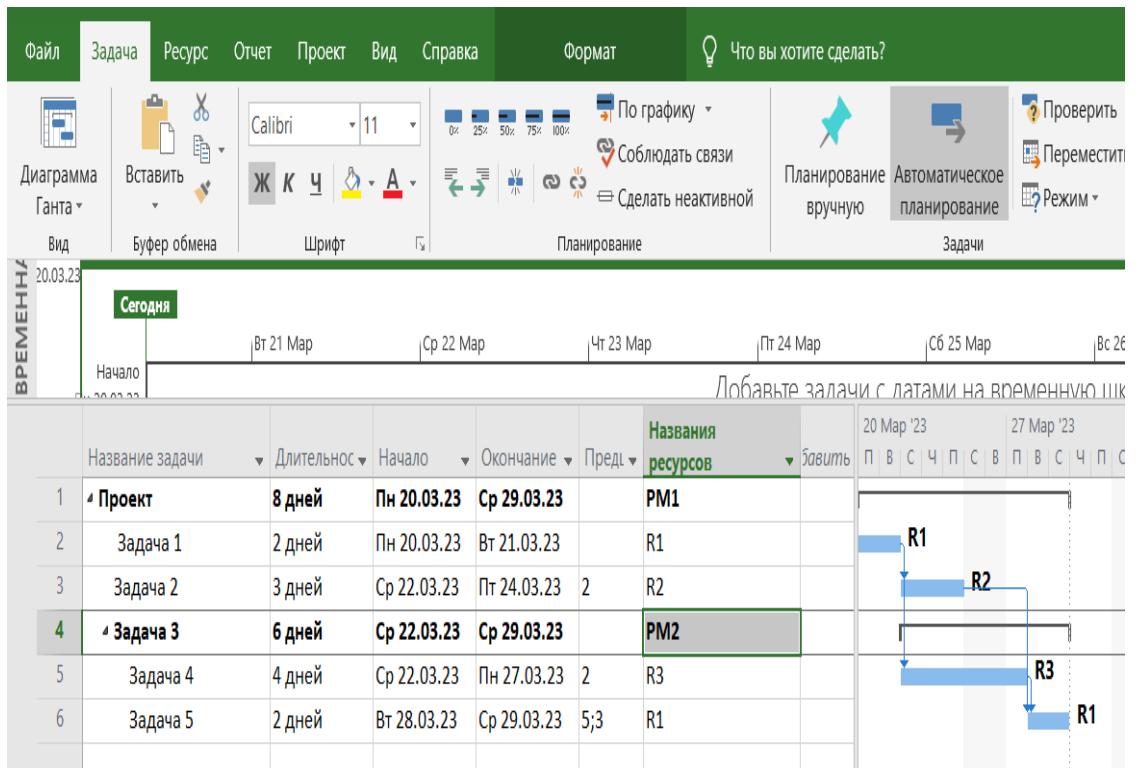


Рис. 7.4 Джерела зв'язків

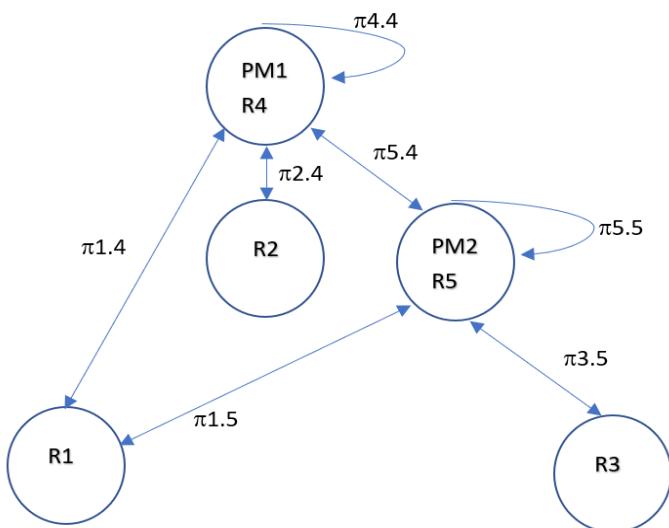


Рис. 7.5 Граф джерел за проектом

В ряді випадків виникає потреба об'єднання інформації графа джерел, побудованому на основі визначених комунікацій за структурою проекту (рис. 7.5) з відповідними комунікаціями, що існують на основі структурної підлегlosti виконавців (рис. 7.6). Таке об'єднання інформаційних комунікаційних зв'язків дає повну картину можливих напрямків обміну інформації у проекті (рис. 7.7).

Зв'язки за підлеглістю виконавців та сумарний граф.

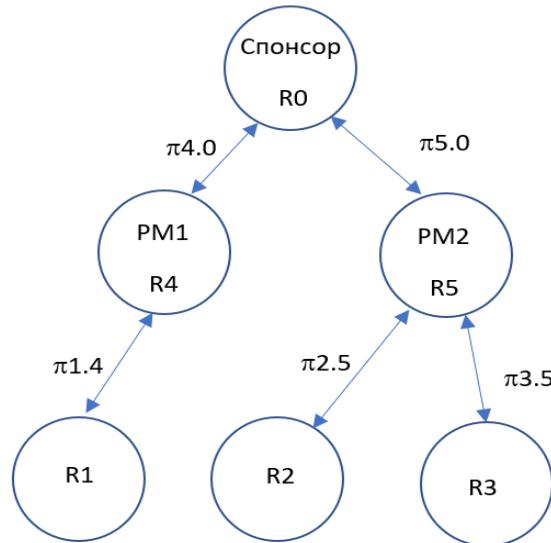


Рис. 7.6 Граф зв'язків за організаційною підлеглістю

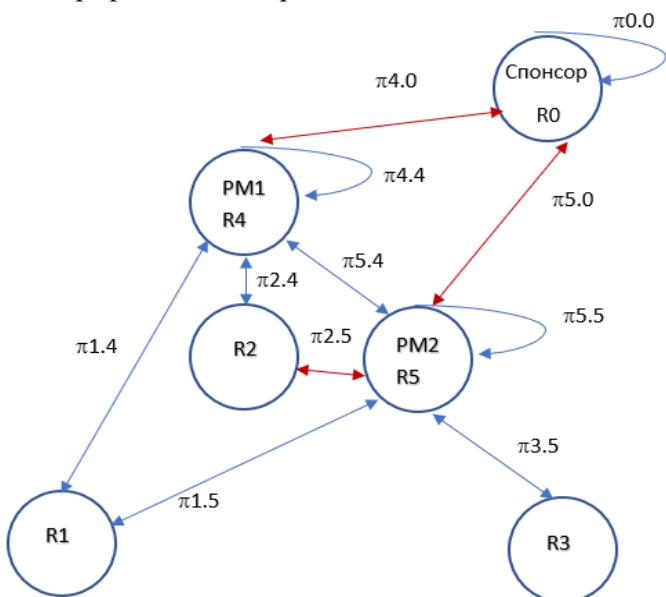


Рис. 7.7 Повний граф зв'язків

Питання до розділу 7

1. Що включає управління комунікаціями проєкту?
2. Сутність інформаційного обміну в організації?
3. Що включає планування системи комунікацій?
4. Які особливості інформаційної системи управління проєктом?
5. Опишіть зв'язки ролей у проєкті?
6. Методи формування комунікацій.

РОЗДІЛ 8. УПРАВЛІННЯ РИЗИКАМИ У ПРЄКТАХ

8.1. Поняття ризику та управління ризиками

Процеси прийняття рішень в управлінні проектами відбуваються, як правило, в умовах наявності тієї чи іншої **міри невизначеності**, яка визначається наступними факторами [1, 7]:

- **неповним знанням всіх параметрів**, обставин, ситуації для вибору оптимального рішення, а також неможливістю адекватного і точного обліку всієї навіть доступної інформації і наявностю імовірнісних характеристик поведінки середовища;
- **наявністю фактору випадковості**, тобто реалізації факторів, які неможливо попередити і спрогнозувати навіть в ймовірній реалізації;
- **наявністю суб'єктивних факторів протидії**, коли прийняття рішень йде в ситуації гри партнерів з протилежними або незбіжними інтересами.

Таким чином, реалізація проекту йде в умовах невизначеності та ризиків, і ці дві категорії взаємопов'язані.

Невизначеність в широкому сенсі - це неповнота або неточність інформації про умови реалізації проекту, в тому числі пов'язаних з ними витрати і результати.

Ризик – потенційна, чисельно вимірна можливість **несприятливих ситуацій** і пов'язаних з ними **наслідків** у вигляді втрат, збитку, збитків, наприклад – очікуваного прибутку, доходу або майна, грошових коштів у зв'язку з невизначеністю, тобто з випадковим зміною умов економічної діяльності, несприятливими, в тому числі форс- мажорними обставинами, загальним падінням цін на ринку; можливість отримання непередбачуваного результату в залежності від прийнятого господарського рішення, дії.

- **«Ймовірність ризиків»** – ймовірність того, що в результаті прийняття рішення **відбудуться втрати** для підприємницької фірми, тобто ймовірність небажаного результату.

- Існують два методи визначення ймовірності небажаних подій: **об'єктивний і суб'єктивний**.

- **Об'єктивний метод** заснований на **обчисленні частоти**, з якою той чи інший результат був отриманий в аналогічних умовах.

- **Суб'єктивна ймовірність** є припущенням щодо певного результату. Цей метод визначення ймовірності небажаного результату заснований на **судженні і особистому досвіді підприємця**.

В даному випадку відповідно до минулого досвіду і інтуїції менеджеру проекту необхідно зробити **оціочне припущення** про імовірність подій.

Вимірювання ризиків – визначення ймовірності настання ризикової події.

Оцінюючи ризики, які в змозі прийняти на себе команда проекту і інвестор проекту при його реалізації, виходять насамперед із специфіки і важливості проекту, з наявності необхідних ресурсів для його реалізації і можливостей фінансування ймовірних наслідків ризиків.

Ступінь допустимих ризиків, як правило, визначається з урахуванням таких параметрів, як розмір і надійність інвестицій в проект, запланованого рівня рентабельності та ін.

У кількісному відношенні невизначеність має на увазі можливість відхилення результату від очікуваного (або середнього) значення як в меншу, так і в більшу сторону.

Відповідно можна уточнити поняття ризику.

Ризик – це **ймовірність втрати частини ресурсів, недоотримання доходів або появи додаткових витрат і (або) зворотне - можливість отримання значної вигоди (доходу)** в результаті здійснення певної цілеспрямованої діяльності. Тому ці дві категорії, що впливають на реалізацію інвестиційного проекту, повинні аналізуватися і оцінюватися спільно.

Управління ризиками проекту (Project risk Management) – це спрямований процес пошуку, прийняття і виконання організаційних та фінансових управлінських рішень, дія яких спрямована на зниження можливості виникнення несприятливих умов для реалізації проекту і мінімізацію можливих незапланованих втрат проекту на етапі реалізації проекту (рис. 8.1).

Управління ризиками є підсистемою управління проектом.

Найбільш поширеними для **ІТ та програмних проектів** є такі категорії ризиків:

- ризик збільшення витрат по проекту (збільшення обсягів робіт);
- ризик затримок виконання робіт (збільшення часу виконання робіт);
- ризик несвоєчасності поставок обладнання;
- ризик затримок платежів від інвестору проекту чи кредитору.

Ці ризики визначають кінцеві терміни та вартість виконання робіт за проектами.

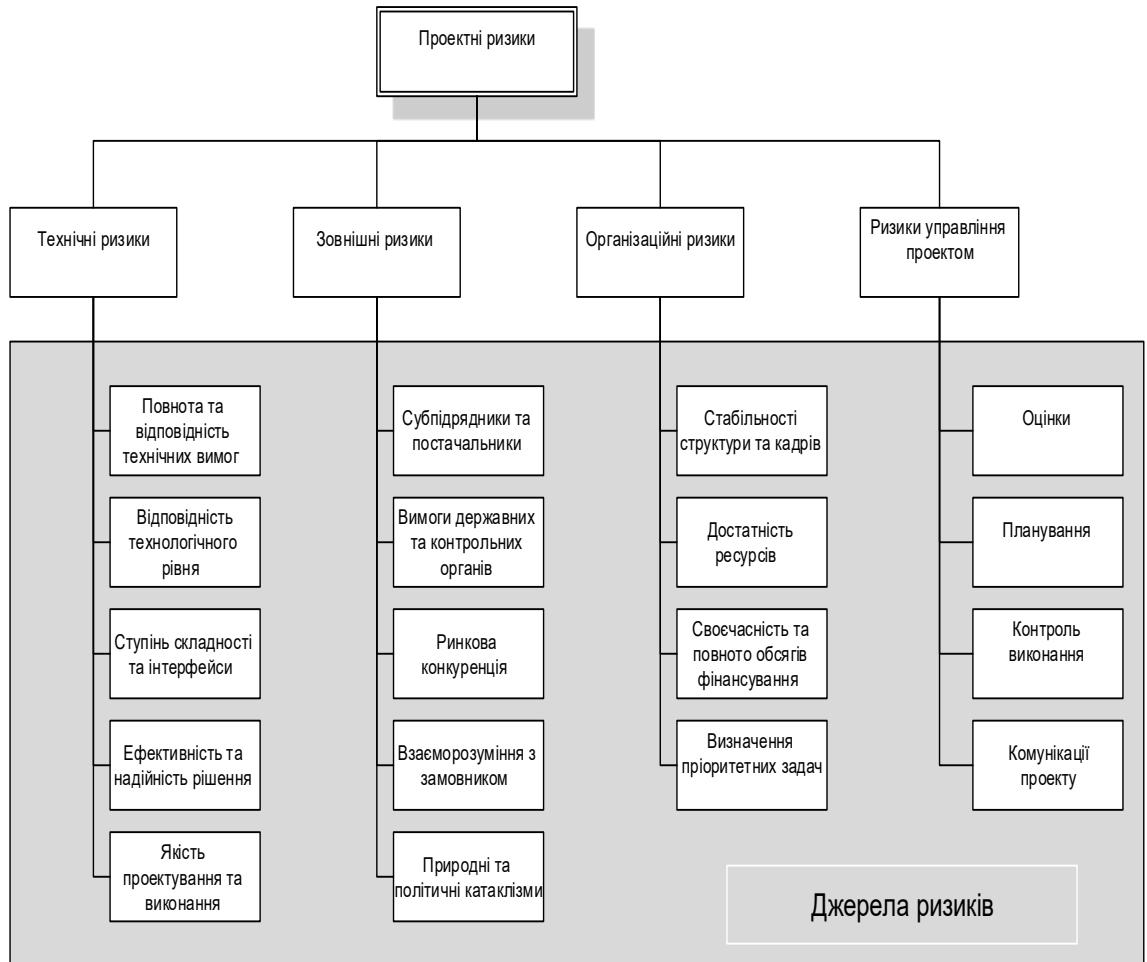


Рис. 8.1 Групи ризиків та джерела їх появи

Управління ризиками проєкту:

- **Виявлення та ідентифікація** передбачуваних ризиків.
- **Аналіз і оцінка** ризиків.
- Вибір методів управління ризиком.
- Застосування обраних методів і прийняття рішень в умовах ризиків.
- **Реагування** на наступ ризикового події.
- Розробка і реалізація **заходів зниження** ризиків.
- Контроль, аналіз та оцінка дій щодо **зниження ризиків**.

Методи управління ризиками:

- Розробка і реалізація **стратегії** управління ризиками.

- **Методи компенсації ризиків**, що включають прогнозування зовнішнього середовища проєкту, маркетинг проєктів і продуктів проєкту, моніторинг соціально-економічного та правового середовища і створення системи резервів проєкту.

- **Методи локалізації ризиків**, які застосовуються для високо ризикових проєктів в багато проєктної системі, що мають на увазі створення окремих спеціальних підрозділів для реалізації особливо ризикових проєктів.

- **Методи уникнення ризиків**, що включають відмову від ризикованих проєктів і ненадійних партнерів, страхування ризиків, пошук гарантів.

Методи аналізу та оцінки ризиків:

- **Аналіз чутливості.**
- Перевірка стійкості.
- Визначення точки беззбитковості.
- Коригування параметрів проєктів.
- **Формалізований опис невизначеності.**
- **Аналіз сценаріїв.**
- Метод Монте-Карло.
- Метод побудови «дерева рішень».

Організація робіт з аналізу ризиків:

- Підбір досвідченої команди експертів.
- Підготовка спеціального запитальника і зустрічі з експертами.
- Вибір техніки аналізу ризиків.
- Встановлення факторів ризиків і їх значимості.
- Створення моделі механізму дії ризиків.
- Встановлення взаємозв'язку окремих ризиків і сукупного ефекту від їх впливу.

- Розподіл ризиків між учасниками проєкту.
- Розгляд результатів аналізу ризиків - зазвичай у формі спеціально.
- Підготовлюваного звіту (доповіді).

Зниження ризиків:

- Розподіл ризиків між учасниками проєкту (передача, відведення, **трансферт частини ризиків** співвиконавцям).

- **Страхування ризиків.**
- **Резервування.**

8.2. Аналіз проєктних ризиків

Аналіз проєктних ризиків підрозділяється на **якісний** (опис всіх передбачуваних ризиків проєкту, а також вартісна оцінка їх наслідків та заходів щодо зниження) і **кількісний** (безпосередні розрахунки змін **ефективності** проєкту в зв'язку з ризиками).

Аналіз проєктних ризиків базується на оцінках ризиків, які полягають у визначені величини (ступеня) ризиків. Методи визначення критерію кількісної оцінки ризиків включають:

- **статистичні методи оцінки**, що базуються на методах математичної статистики, тобто дисперсії, стандартного відхилення, коефіцієнт варіації. Для застосування цих методів необхідний досить великий обсяг вихідних даних, спостережень;

- **методи експертних оцінок**, засновані на використанні знання експертів в процесі аналізу проєкту і врахування впливу якісних факторів;

- **методи аналогій**, засновані на аналізі подібних проєктів і умов їх реалізації для розрахунку ймовірностей втрат. Ці методи застосовуються тоді, коли є представницька база для аналізу та інші методи неприйнятні або менш достовірні. Ці методи широко використовуються на Заході, оскільки в практиці управління проєктами використовуються оцінки проєктів після їх завершення і накопичується значний матеріал для подальшого застосування;

- **комбіновані методи** включають використання відразу декількох методів.

8.3. Методи зниження ризиків

1. **Диверсифікація, або розподіл ризиків** (розподіл зусиль підприємства між видами діяльності, результати яких безпосередньо не пов'язані між собою), що дозволяє розподілити ризики між учасниками проєкту. Розподіл проєктних ризиків між його учасниками є ефективним способом їх зниження. Теорія надійності показує, що зі збільшенням кількості паралельних ланок в системі ймовірність відмови в ній знижується пропорційно кількості таких ланок. Тому розподіл ризиків між учасниками підвищує надійність досягнення результату. Логічно при цьому зробити відповідальним за конкретний вид ризику того його учасника, який має можливість точніше і якісніше розраховувати і контролювати цей ризик. Розподіл ризиків оформлюється при розробці фінансового плану проєкту та контрактних документів.

Розподіл ризиків фактично реалізується в процесі підготовки плану проєкту та контрактних документів. Слід мати на увазі, що підвищення ризиків у

одного з учасників має супроводжуватися адекватним зміною в розподілі доходів від проєкту.

Тому при переговорах необхідно:

- визначити можливості учасників проєкту по зменшенню впливу настання ризикових подій;
- визначити ступінь ризиків, яку бере на себе кожен учасник проєкту;
- домовитися про прийнятний винагороду за ризики;
- стежити за дотриманням паритету в співвідношенні ризиків і доходу між усіма учасниками проєкту.

2. Резервування коштів на покриття непередбачених витрат являє собою спосіб боротьби з ризиком, який передбачає встановлення співвідношення між потенційними ризиками, які впливають на вартість проєкту, і розміром витрат, необхідних для подолання збоїв у виконанні проєкту.

Величина резерву повинна бути дорівнювати або перевищувати величину коливання параметрів системи в часі. У цьому випадку витрати на резерви повинні бути завжди нижче витрат (втрат), пов'язаних з відновленням відмови. Зарубіжний досвід допускає збільшення вартості проєкту від 7 до 12% за рахунок резервування коштів на форс-мажор. Резервування коштів передбачає встановлення співвідношення між потенційними ризиками, що змінюють вартість проєкту, і розміром витрат, пов'язаних з подоланням порушень в ході його реалізації.

3. Страхування ризиків. У разі, якщо учасники проєкту не в змозі забезпечити реалізацію проєкту при настанні тієї чи іншої ризикової події власними силами, необхідно здійснити страхування ризиків.

Страхування ризиків є по суті передача певних ризиків страховій компанії.

Зарубіжна практика страхування використовує повне страхування інвестиційних проєктів. У нас умови дійсності дозволяють поки тільки частково страхувати ризики проєкту: будівлі, устаткування, персонал, деякі екстремальні ситуації.

Ефективність методів зниження ризиків визначається за допомогою наступного алгоритму:

- розглядається ризик, що має найбільшу **важливість** для проєкту;
- **визначається перевитрата коштів** з урахуванням ймовірності настання несприятливої події;
- **визначається перелік можливих заходів**, спрямованих на зменшення ймовірності та небезпеки ризикової події;
- **визначаються додаткові витрати** на реалізацію запропонованих заходів;

- **порівнюються необхідні витрати** на реалізацію запропонованих заходів з можливою перевитратою коштів внаслідок настання ризикової події;
- **приймається рішення про здійснення або про відмову** від заходів по усуненню наслідків ризиків;
- процес зіставлення ймовірності і наслідків ризикових подій з витратами на заходи щодо їх зниження повторюється **для наступного за важливістю ризику**.

8.4. Автоматизовані методи оцінки ризиків

Імовірність виникнення ризиків має відображати сукупні **середні відносні порушення строків** по всіх задачах, що виконуються на момент оцінки цього впливу [4].

Вона може бути оцінена та визначена по впливу ризику **на момент оцінки цього впливу шляхом** розрахунку **середнього по всіх задачах порушення строків виконання задач**, що виконуються на момент оцінки, у процентах.

Визначимо значення як (рис. 8.2):

k – кількість задач, що виконуються на момент оцінки ризику;

dpi – дата початку i – ої задачі, що виконується на момент оцінки ризику;

dzi – дата завершення i – ої задачі, що виконується на момент оцінки ризику;

dvi – дата внесення даних про виконання i – ої задачі, що виконується на момент оцінки ризику;

doi – дата оцінки ризику;

rvi – процент виконання i – ої задачі на дату внесення даних.

Процент виконання i – ої задачі на дату оцінки ризику, що визначено внесеними даними, може бути визначено як:

$$poi = rvi * (doi - dpi) / (dvi - dpi).$$

Процент виконання i – ої задачі на дату оцінки ризику, що прогнозується за планом може бути визначено як:

$$pni = (doi - dpi) / (dzi - dpi) * 100.$$

Тоді

$$x = \sum_{i=1}^k (pni - poi) / k.$$

Тільки для тих робіт де $poi < pni$.

Класифікація ризиків по імовірності виникнення може бути визначена на основі поділу на п'ять рівнів, що досить детально відображає різні можливі ситуації у ході аналізу ризиків. Приклад такого поділу наведено у таблиці 8.1, де визначена бальна

оцінка імовірності виникнення ризику по результатам оцінки інформації про виконання задач проекту.

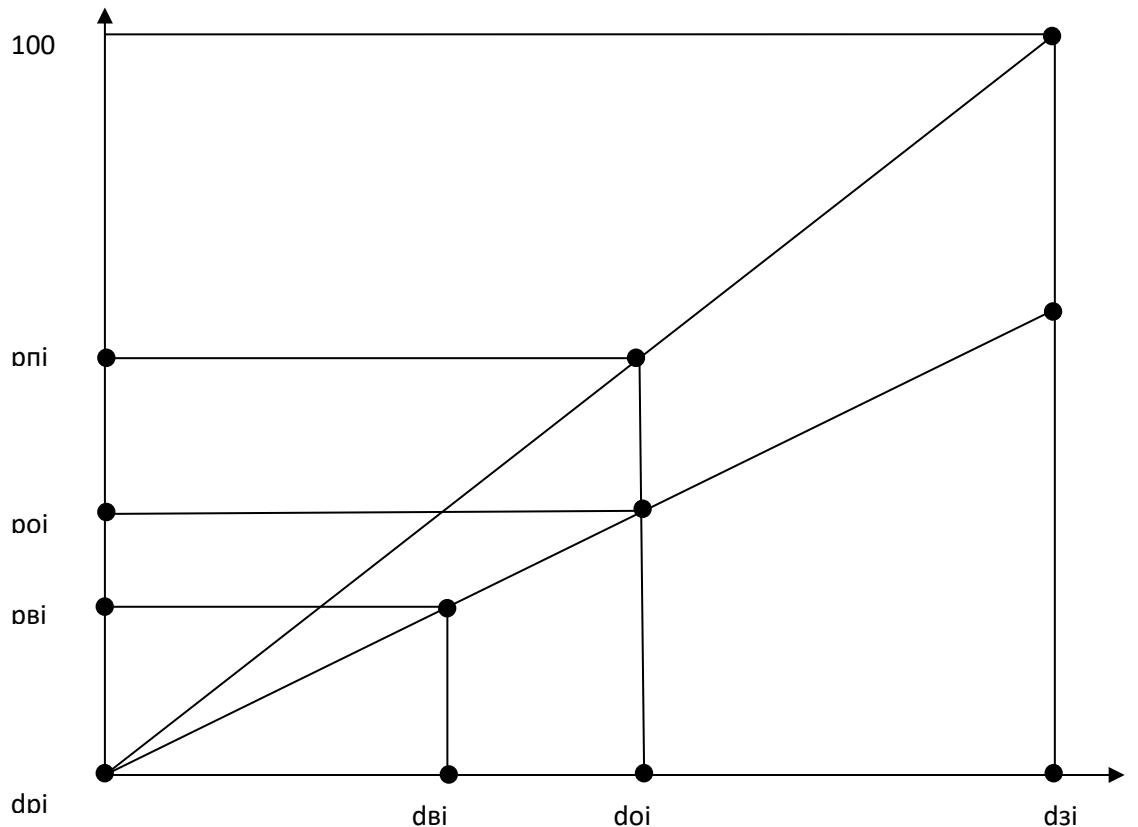


Рис. 8.2 Визначення значень процентів виконання задач проекту за внесеними даними та планом

Таблиця 8.1 Бальна оцінка імовірності виникнення ризиків по результатам визначення стану виконання проекту

Імовірність виникнення	
I (бали)	(в %)
1	$0\% < x \leq 10\%$
2	$10\% < x \leq 30\%$
3	$30\% < x \leq 60\%$
4	$60\% < x \leq 90\%$
5	$90\% < x \leq 100\%$

Величина втрат має відображати **загальне відносне порушення** строків по всьому проєкту у цілому. Оцінка розміру втрат може бути виконана шляхом визначення значення найбільшого з можливих впливів по окремих задачах, що виконуються на час виконання оцінки ризику, на кінцевий строк виконання усього проєкту.

Оцінку можливих втрат у процентах на основі аналізу результатів стану проєкту на момент оцінки розміру ризику можливо визначити наступним шляхом:

$$y = \max((pni - poi) * (dzi - dpi)) / dp$$

де, dp – планова тривалість усього проєкту.

Тільки для тих робіт де $poi < pni$.

На основі визначеного значення y проводиться бальна оцінка величини можливих втрат від ризику по результатам визначення стану виконання задач проєкту.

Таблиця 8.2 Бальна оцінка величини втрат від ризику по результатам визначення стану виконання проєкту

Величина втрат	
B (бали)	(в %)
1	$0\% < y \leq 10\%$
2	$10\% < y \leq 30\%$
3	$30\% < y \leq 60\%$
4	$60\% < y \leq 90\%$
5	$90\% < y \leq 100\%$

Для оцінки можливого впливу ризику на стан виконання проєкту може бути використане значення показника індексу ризику.

Індекс ризику – це показник оцінки вірогідних втрат в балах, який дає можливість комплексно оцінити ступінь дії і рівень загрози ризику.

Оцінка індексу ризику проводиться за формулою:

$$R = I * B,$$

де: R – індекс ризику

$I=f1(x)$ – ймовірність виникнення ризиків, відповідно до таблиці 8.1 (в балах).

$B=f2(y)$ – величина втрат, відповідно до таблиці 8.2 (в балах).

$f1, f2$ – задані таблична функції, що визначають перехід від обчислених на основі даних про виконання проєкту значень оцінок x та y , до ціличисельних бальних оцінок.

Процес оцінки в ході реалізації проектів базується на оцінці ступеня дії ризиків по кожному з можливих ризиків згідно поточному значенню індексу ризиків (значення R для кожного з ризиків на поточний час аналізу стану виконання проекту).

В залежності від отриманого значення індексу ризиків для кожного з можливих ризиків проводиться оцінка необхідності та форми реагування на вплив ризику (таблиця 8.3). В обґрунтovаних випадках оцінки вартості або тривалості задач проекту можуть бути скореговані на величину пов'язаних з цими задачами ризиків.

Таблиця 8.3 Вплив ризику та реакція на його вплив

Індекс ризику (R)	Ступінь впливу ризику	Категорії ризиків	Реакція на вплив ризику
$1 \leq R \leq 4$	Невпливовий. Відсутність будь-якого впливу на хід реалізації проекту	Прийняті ризики. Визначаються, як ризики, що не потребують термінової реакції, але можуть бути занотовані для подальшого аналізу	Прийняття ризику. Передбачають прийняття ризику. Прийняття може бути активним (при попередній розробці заходів на випадок настання події ризику) або пасивним, при якому достатньо просто передбачити в плані грошові, часові резерви або занижений прибуток
$5 \leq R \leq 8$	Незначний вплив. Збільшення тривалості виконання робіт, об'єми додаткових робіт в рамках бюджету і планових термінів завершення; виробничі дефекти незначні	Виправдані ризики. Вторинні для обробки. Кожний ризик з цим рівнем повинен враховуватися під час виконання робіт. Наявність таких ризиків вимагає виконання певних дій, що вплинуть на зменшення цього ризику	Зменшення впливу. Потребують зменшення впливу ризику через зменшення вірогідності події ризику. Цього можна досягти, наприклад, заходами Профілактики, що зменшить вірогідність ризику. Можна зменшити втрати від ризику шляхом страхування, дублювання або іншими заходами
$9 \leq R \leq 10$	Помірний вплив Збільшення тривалості виконання робіт, помітні виробничі дефекти, недотримання технічних рішень, об'єми додаткових	на кінцеві результати виконання проекту. Ризик повинен знаходитися під постійним контролем і його рівень повинен періодично переоцінюватися	

Кінець таблиці 8.3

Індекс ризику (R)	Ступінь впливу ризику	Категорії ризиків	Реакція на вплив ризику
	робіт вимагають узгоджень зі Спонсором проекту		
$12 \leq R \leq 16$	Істотний вплив. Збільшення тривалості виконання робіт, виробничий брак, недотримання технічних рішень, об'єми додаткових робіт недопустимі для Спонсора	Неприпустимі ризики. Первінні для обробки. Дії по усуненню впливу таких ризиків мають бути першочерговими. Зниження їх впливу, як правило, потребує втручання керівництва для залучення додаткових ресурсів по виконанню проєкту (трудових, фінансових, матеріальних та ін.)	Усунення ризику. Потребують усунення ризику шляхом ліквідації його потенційної причини. В деяких випадках це можливо. Наприклад, виключити ризик різного розуміння вимог до якості замовником і постачальником можна, чітко сформувавши ці вимоги в тексті контракту
$20 \leq R \leq 25$	Критичний вплив Крайній ступінь порушення плану та технічних вимог до проєкту		

Такий підхід до оцінки ступеню впливу ризиків на кінцеві результати виконання проєктів дозволяє достатньо просто реалізувати його як надбудову до стандартного програмного забезпечення.

Важливо не тільки виявити потенційні ризики проєкту, але і оцінити їх вплив на результати, своєчасно прийняти рішення про зниження ризиків, причому здійснювати управління ризиками на всіх стадіях реалізації проєкту і адекватно задокументувати процеси управління ризиками проєкту для подальшого застосування цих знань у подальшій практиці управління подібними проєктами.

Питання до розділу 8

- Що таке управління ризиками проєкту?
- Як визначається ризик у проєкті?

3. Що таке імовірність ризику?
4. Які є напрямки боротьби з проектними ризиками?
5. Які є категорії ризиків?
6. Що включає управління ризиками проекту?
7. Як можна класифікувати ризики по імовірності?

РОЗДІЛ 9. УПРАВЛІННЯ ВАРТІСТЮ ТА БЮДЖЕТОМ

9.1. Процеси управління вартістю проєкту

Управління вартістю проєкту (Project Cost Management) – розділ проектного менеджменту, що операє процесами, необхідними для забезпечення дотримання бюджету проєкту.

Управління вартістю проєкту об'єднує процеси, що виконуються *в ході планування, розробки бюджету залучення фінансування, фінансування, управління та контролю вартості*, що забезпечують виконання проєкту у рамках затвердженого **бюджету проєкту**.

Управління вартістю проєкту включає наступні процеси:

- **планування ресурсів проєкту** – визначення того, які ресурси і в яких кількостях необхідні для виконання робіт проєкту;
- **оцінка вартості ресурсів** – розробка приблизної оцінки вартості ресурсів, необхідних для виконання робіт з проєкту;
- **розробка бюджету проєкту** – складання кошторису для кожного виду роботи з проєкту;
- **аналіз ресурсів проєкту** – оцінка відхилень витрат й продуктивності використовуваних ресурсів від планових значень;
- **контроль бюджету проєкту** – контроль над змінами в бюджеті проєкту.

Управління вартістю проєкту має враховувати **вимоги до інформації про витрати**, що пред'являються зацікавленими сторонами проєкту. *Різні зацікавлені сторони* проєкту можуть розраховувати вартість проєкту *різними способами і в різні моменти часу*.

Управління вартістю проєкту стосується, насамперед, **вартості ресурсів**, необхідних для виконання операцій проєкту.

Крім того, при управлінні вартістю проєкту слід враховувати, як прийняті рішення можуть позначатися на **подальших періодичних витратах на експлуатацію**, обслуговування та **технічну підтримку продукту**, послуги або результату проєкту.

9.2. Планування ресурсів проєкту

Ресурси проєкту – це все те, що необхідно для виконання операцій (задач) проєкту.

Ресурси поділяються на IT-проєкти:

- **трудові ресурси;**
- обладнання;
- приміщення;
- **грошові засоби;**
- енергетичні ресурси;
- **інформаційні ресурси;**
- **обчислювальна та оргтехніка;**
- **програмне забезпечення;**
- **мережеві ресурси.**

Ресурси можуть бути:

- **поновлюваними** (типу «потужності», називають просто ресурсами) – це люди, матеріали й механізми, які після виконання операції можуть бути використані знову. Вони відновлюються, не нагромаджуються і не накопичуються.

Якщо ці ресурси не використовуються, то їх функціональна здатність в даний проміжок часу не може бути компенсована в майбутньому, не може бути нагромаджена.

- **непоновлюваними** (типу «енергія», називають ще матеріалами) – це матеріали й устаткування, які на операціях витрачаються. Такі ресурси не відтворювані, накопичувальні, складовані, які витрачаються повністю, не допускаючи повторного використання.

Планування ресурсів означає визначення того, **які ресурси** (люди, устаткування і матеріали) і **в якій кількості будуть використані** на роботах проекту.

Планування ресурсів – ітеративний процес.

Цей процес тісно **пов’язаний з плануванням операцій**, плануванням вартості й складанням розкладу виконання проекту, за результатами яких результати планування ресурсів можуть переглядатися.

Процеси управління ресурсами призначаються для **внесення змін у затверджений план виконання проекту** – або за результатами оцінки виконання, якщо супровідні зміни не приводять до необхідності затвердження змін у спонсора або для реалізації затверджених запитів на зміни.

Вхідна інформація планування ресурсів проекту. Констатація цілей. Констатація цілей містить обґрутування і мету проекту, що важливо враховувати при плануванні ресурсів.

WBS. Ієрархічна структура робіт визначає ті елементи проекту, які мають потребу у використанні ресурсів.

Перелік операцій. Операції проєкту – це той нижній рівень WBS, на якому використовуються ресурси.

Ресурси повинні призначатися на всі операції, для виконання яких вони потрібні. З іншого боку, у процесі планування ресурсів часто виникає необхідність перегляду операцій.

Потреби операцій у ресурсах. Для того, щоб визначити, які ресурси призначити на виконання операцій проєкту, треба виділити ті види ресурсів, які здатні виконати роботи, передбачені на кожній з операцій.

Обсяги робіт на операціях. Обсяги робіт на операціях необхідно враховувати при виборі використовуваних ресурсів. Так, немає рації використати високопродуктивне устаткування при малих обсягах робіт.

Продуктивність ресурсів. Продуктивність ресурсів відіграє важливу роль при призначенні ресурсів на операції проєкту. Вони звичайно задаються в тих же одиницях, але віднесених до часу, в якому вимірюється обсяг робіт.

Політика і структура організації. Мається на увазі політика закупівлі матеріалів і устаткування, проведення підрядних торгов та ін.

Історична інформація. Під історичною розуміється інформація про типи ресурсів, використаних в проєктах і на аналогічних операціях у минулому.

Обмеження. Обмеження можуть визначатися як перерахованими вище чинниками, так і зовнішніми умовами – погодними законодавчими та ін.

Використовувані методи і засоби. Експертні оцінки. Експертні оцінки часто необхідні при плануванні ресурсів. Для експертизи можна запросити фахівців з інших підрозділів виконуючої організації, консалтингових компаній, професійних асоціацій.

Вихідна інформація планування ресурсів проєкту. Призначення ресурсів на операції проєкту. Основним виходом процесу планування ресурсів є перелік типів і кількості ресурсів, необхідних для виконання всіх елементів WBS.

Ці ресурси будуть уточнюватися за результатами наступних стадій планування (складання розкладу виконання проєкту й планування вартості) і аналізу плану.

Календарі ресурсів. Конкретизація використовуваних у проєкті ресурсів дозволяє визначити і їхні календари. Наприклад, 5-денний робочий тиждень з 8-годинним робочим днем і встановленими відпустками.

9.3. Оцінка вартості операцій

Вартості ресурсів. Вартості ресурсів можуть визначатися по-різному. Для поновлюваних ресурсів звичайно задається вартість години їхньої роботи, для матеріалів – вартість одиниці.

Для підрахунку вартості операції через вартості години роботи ресурсів необхідно знати тривалість роботи ресурсів на цій операції. Крім того, слід враховувати, що поновлювані ресурси в процесі своєї роботи можуть витрачати матеріали, вартість яких повинна враховуватися при підрахунку вартості операції проєкту.

Умови контрактів. Вартості ресурсів на окремих операціях проєкту можуть визначатися умовами контрактів. Крім того, вартість години роботи ресурсу може також задаватися неоднозначно. У проєкті може виявится необхідним використати персонал у вихідні дні або подовжити тривалість робочого дня в деякі періоди. Вартість години роботи ресурсу при подібних змінах відрізняється від звичайної і визначається умовами контрактів з персоналом.

Вартості операцій. Вартості операцій можуть визначатися контрактом або складатися з таких складових:

- постійної складової вартості операції;
- постійної складової вартості призначень ресурсів на операцію;
- вартості роботи поновлюваних ресурсів.

При попередній оцінці вартості операцій проєкту часто використовують нормативи. Звичайно нормується вартість одиниці об’єму, тому для підрахунку вартості операції необхідно знати і обсяг робіт на операції. Такі оцінки застосовують також при визначенні й узгодженні контрактної ціни. При такому підході вартість ресурсів не враховується.

Історична інформація. Вартості ресурсів і операцій при нестачі інформації можуть визначатися за аналогами, для чого є необхідна історична інформація. Вона включає:

- файли колишніх проєктів;
- комерційні бази даних;
- знання команди проєкту.

Структура витрат. Треба задати ієрархічну структуру й систему кодування витрат, які потрібні для подання необхідної вартісної звітності.

WBS. *Ієрархічна структура робіт* тут необхідна для контролю того, щоб всі операції проєкту мали вартісну оцінку.

Для оцінки вартості операцій використовують наступні методи.

Оцінка за аналогами. Оцінку за аналогами ще називають оцінкою «зверху вниз». Це означає використання вартості попередніх аналогічних проектів як основи для оцінки вартості поточного проекту. Оцінка за аналогами часто використовується тоді, коли не вистачає детальної інформації про проект. Оцінка за аналогами – це різновид експертної оцінки.

Оцінка за аналогами менш трудомістка, ніж інші методи, але вона й менш точна. На неї можна покладатися, коли не тільки попередні проекти були дійсно аналогічні, але й коли особи, які готували оцінку, мають відповідний досвід.

Параметричне моделювання. Параметричне моделювання означає використання характеристик проекту в математичній моделі, призначеної для оцінки вартості проекту. Моделі можуть бути простими (оцінка вартості житла через вартість квадратного метра) або складними з використанням великої кількості чинників.

На трудомісткість і точність параметричного моделювання впливають такі чинники:

- точність використованої історичної інформації;
- вимірність використовуваних параметрів;
- масштабність моделі (можливість використання результатів для проектів різного розміру).

Оцінка «знизу вгору». Цей метод означає оцінку вартості операцій проекту, а потім підсумовування цих вартостей для оцінки вартості всього проекту.

Точність і трудомісткість такої оцінки визначається ступенем деталізації робіт проекту. Чим більш детально проект розбитий на операції, тим вище як трудомісткість, так і точність оцінки «знизу вгору». Команда проекту повинна знайти оптимальне співвідношення між трудомісткістю й точністю.

Треба, однак, відзначити, що коли оцінка вартості проекту припускає облік інфляції, повернення відсотків по кредитах та інших чинників, обумовлених розподілом витрат у часі, вартісну оцінку проекту в цілому слід робити на базі складеного розкладу виконання проекту.

Програми управління проектами. Програми управління проектами широко використовують для оцінки вартості. Використання підходящеї програми значно знижує трудомісткість таких оцінок і дозволяє швидко розрахувати різні альтернативні варіанти. Треба тільки уважно поставитися до вибору використованої програми – у різних програм управління проектами можливості обліку різних складових вартості операцій і ресурсів проекту відчутно відрізняються.

Оцінка вартості. Під оцінкою вартості розуміється призначення очікуваної вартості всім операціям, ресурсам і призначенням проєкту. Крім того, оцінка вартості включає оцінку очікуваної інфляції, кредитних відсотків, дисконту та інших додаткових показників, які необхідно враховувати при вартільному аналізі проєкту. Перелік цих показників залежить від конкретного проєкту. Вартісні оцінки повинні задаватися в тій валюті, в якій передбачаються витрати, для того щоб полегшити перерахування при змінах обмінного курсу.

Додаткова інформація. Додаткова інформація оцінки вартості повинна включати:

- опис оціненої роботи, для якого часто досить посилання на WBS;
- опис використаних методів оцінки;
- опис всіх використаних припущень і допущень;
- точність оцінок.

Перелік додаткової інформації залежить від області використання. Але навіть грубий опис дозволяє краще зрозуміти, як були розроблені вартісні оцінки.

9.4. Розробка бюджету проєкту

Розробка бюджету (рис. 9.1) – це є визначення базисної лінії вартості проєкту, що показує розподіл у часі нарastaючим підсумком витрат за проєктом і служить для порівняння поточних результатів з плановими.

Розробка бюджету – головна складова бюджетування проєкту, під яким розуміється визначення вартісних показників у рамках проєкту робіт і проєкту в цілому, процес формування бюджету проєкту, що містить установлений (затверджений) розподіл витрат за видами робіт, статтями витрат, часом виконання робіт, центрами витрат або за іншою структурою.

Бюджет проєкту містить у собі сумарні оцінні витрати, необхідні для реалізації проєкту. Перш ніж приступати до реалізації проєкту, треба визначити вимоги по обсягах робіт і вимоги до бюджету. Ці чинники є вкрай важливими, оскільки являють собою цільовий план, з яким порівнюється виконання проєкту.

Протягом усього життєвого циклу проєкту слід контролювати показники фінансування, порівнювати їх з плановими і при необхідності вносити виправлення й зміни. По закінченні проєкту визначається освоєння витрат, для чого фактичні витрати порівнюються зі значеннями, закладеними в бюджеті.



Рис. 9.1 Планування бюджету проекту

У зв'язку з **обмеженістю бюджету** проекту з неминучістю виникають **конфлікти між підрозділами**, залученими в реалізацію проекту, за фінансові ресурси. На більш високому рівні ієархії організації (підприємства) виникають конфлікти між проектами й групами проектів.

Проект має виконуватися в рамках установленого бюджету. У разі перевищення бюджету проект може бути закритий або припинений. Отже однією з необхідних умов здійсненості проекту поряд з плануванням і контролем змісту проекту, визначенням складу робіт є здійснення точного фінансового планування і контролю (бюджетування).

У ринкових умовах саме система бюджетування проекту стає основою його планування. Вся система планування проекту повинна будуватися на основі бюджетування, тобто всі витрати й результати повинні мати чітко фінансове вираження.

В остаточному підсумку одним з основних завдань фінансового планування (бюджетування) проекту є складання балансової моделі, що дозволяє оцінити динаміку балансових даних, плану прибутків і збитків, руху грошових коштів, найважливіших показників рентабельності, оборотності та інших умов як по окремих проектах, так і по підприємству в цілому.

Складання бюджету «зверху вниз» включає визначення витрат на проект на верхньому рівні. Звичайно подібне визначення витрат виконується керівництвом, відповідальним за матеріальні активи, або групою планування витрат, що виконує схожі функції.

Мета складання бюджету «зверху вниз» – довгострокове планування. Як правило, бюджети, що складаються зверху вниз, не враховують деталей проектів і тому не можуть дати точного визначення витрат.

Складання бюджету «знизу вгору» починається з планування бюджетів окремих компонентів проекту, що перебувають на нижчих рівнях, і наступного об'єднання цих бюджетів на більш високому рівні. Подібні процеси звичайно виконуються керівниками проекту або відповідальними за формування графіка проекту, які, як правило, витрачають багато часу на збір і обробку деталізованої інформації, але й одержувані ними результати мають більш високу точність. З метою більш ефективного планування витрат використовується комбінація складання бюджету як *зверху вниз, так і знизу вгору*.

Розклад виконання проекту. Розклад виконання проекту необхідний для визначення розподілу планових витрат у часі, а також для обліку інфляції, повернення кредитів, дисконтування та інших залежних від часу параметрів, використовуваних у проекті. Крім цього, як вхідна інформація для планування витрат з проекту застосовується:

- кошторисна документація;
- попередній графік грошових потоків (складають при розробці ТЕО);
- аналіз фінансового стану інвестора.

Використовувані методи і засоби. При розробці бюджету використовують ті ж методи, що й при оцінці вартості: оцінка за аналогами, параметричне моделювання, оцінка «знизу вгору», програми управління проектами.

Вихідна інформація розробки бюджету проекту. *Бюджет проекту* – це план, виражений у кількісних показниках, який відзеркалює витрати, необхідні для досягнення поставленої мети. У бюджеті представлені оцінні результати відкоригованого календарного плану й стратегії реалізації проекту. Бюджет проекту підраховують підсумуванням оцінок вартостей за періодами.

9.5. Аналіз ресурсів проекту

Аналіз ресурсів призначений для оцінки відхилень витрати та продуктивності використовуваних ресурсів від планових значень і ухвалення

рішення про необхідність коригування плану проєкту й застосування коригувальних впливів для усунення або зменшення небажаних наслідків виниклих відхилень.

Вхідна інформація аналізу ресурсів проєкту. *Облік виконання*. Облік виконання постачає команду проєкту інформацією про фактичну вартість виконання і фактичну витрату ресурсів на операціях проєкту.

Запити на зміни. Аналіз ресурсів виконується також при надходженні запитів на зміни потреби в ресурсах і строків виконання робіт від учасників проєкту. Запити на зміни потреби в ресурсах і строків виконання окремих операцій і фаз повинні аналізуватися з погляду їхнього впливу на виконання інших операцій, фаз і проєкту в цілому.

Обмеження. При аналізі ресурсів, як і при плануванні, слід враховувати кількісні, календарні, об'ємні та інші обмеження, що впливають на можливості використання ресурсів.

Використовувані методи і засоби. *Система управління ресурсами*. Система управління ресурсами включає методи, процедури й документи для внесення змін у планові потреби в ресурсах і строки виконання робіт при відхиленнях фактичних величин потреб і продуктивності ресурсів від запланованих.

Оцінка виконання. Оцінка фактичної й прогнозної витрати матеріалів виконується аналогічно до того, як оцінюються відповідні вартісні параметри. Оцінка продуктивності поновлюваних ресурсів здійснюється шляхом порівняння фактичних строків виконання робіт із запланованими та аналізу отриманих трендів.

Перепланування. Перепланування при нових (прогнозних) вихідних даних дозволяє оцінити необхідність застосування коригувальних впливів.

Вихідна інформація аналізу ресурсів проєкту. *Скоригований план виконання* операцій проєкту приймається відповідно до прийнятих та затверджених корегувальних змін.

Авторизація змін. Локальні зміни потреб у ресурсах і строків виконання операцій і фаз, що не призводять до необхідності коригування цілей і базового плану. Наприклад, зміни умов контрактів можуть бути санкціоновані командою проєкту. Якщо ж отримані запити на зміни вимагають прийняття коригувальних впливів, то рішення по них приймається у процесі управління.

Досвід роботи. Причини відхилень потреб у матеріалах і продуктивності поновлюваних ресурсів повинні документуватися, включатися в архів проєкту й історичну базу даних для використання як у поточному проєкті, так і в інших проєктах виконуючої організації.

9.6. Контроль бюджету проекту

Аналіз вартості призначений для оцінки відхилень фактичної вартості виконання операцій від планових і ухвалення рішення про необхідність коригування плану проекту й застосування коригувальних впливів для усунення або зменшення небажаних наслідків відхилень.

Основні завдання бюджетного контролю (рис. 9.2):

- одержання точних оцінок витрат;
- розподіл витрат у часі;
- підтвердження витрат;
- своєчасна звітність по витратах;
- виявлення помилкових витрат;
- підготовка звіту про фінансовий стан проекту;
- прогноз витрат.

Система бюджетного контролю.

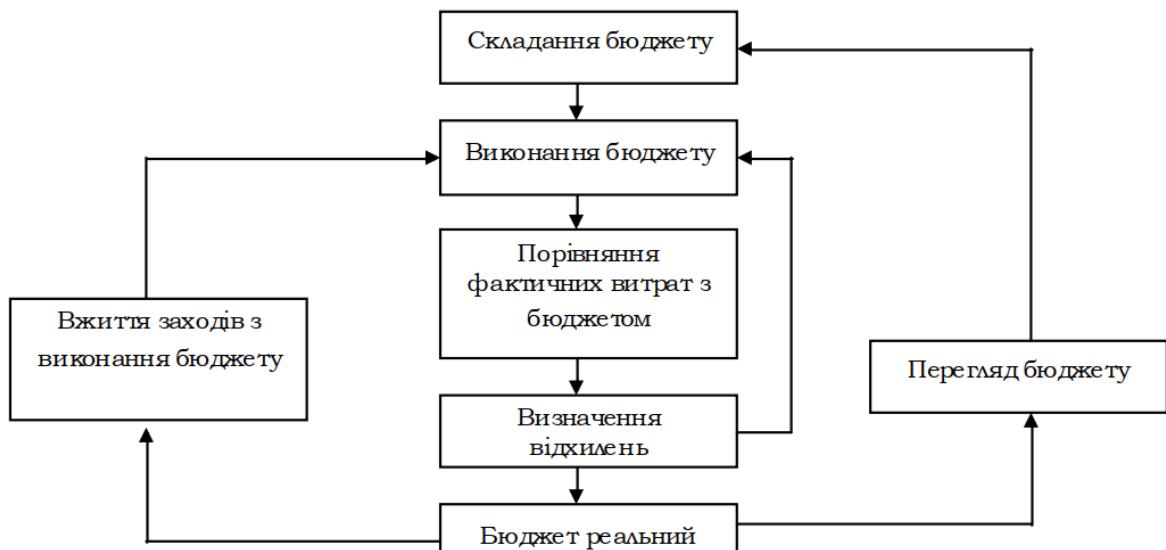


Рис. 9.2 Контроль бюджету

Вхідна інформація контролю бюджету проекту. Запити на зміни. Аналіз вартості виконується також при надходженні запитів на зміни вартості від учасників проекту. Запити на зміни вартості окремих операцій і фаз повинні аналізуватися з погляду їхнього впливу на виконання інших операцій, фаз і проекту в цілому.

Використовувані методи і засоби. Система управління вартістю. Система управління вартістю включає методи, процедури й документи, необхідні для внесення

змін у планові вартості робіт при відхиленнях фактичних величин вартості від запланованих.

Оцінка виконання. Оцінка виконання призначена для ухвалення рішення про необхідність коригувальних впливів. Вона включає вартісний аналіз за методикою освоєного обсягу C/SCSC (Cost/Schedule Control Systems Criteria).

Аналіз C/SCSC у світі використовується найбільш широко. Він поєднує аналіз цілей, вартості й строків і допомагає команді проєкту оцінити хід виконання проєкту. Методики C/SCSC базуються на трьох показниках, які визначаються для кожної операції:

- **Планова вартість запланованих робіт (ПВЗР)** – частина планової вартості операції, що повинна була бути витрачена до розглянутого моменту відповідно до базового плану.

- **Фактична вартість виконаних робіт (ФВВР)** – фактичні витрати на виконані до розглянутого моменту роботи операції.

- **Планова вартість виконаних робіт (ПВВР)** – планова вартість фактично виконаних робіт операції.

Вартість

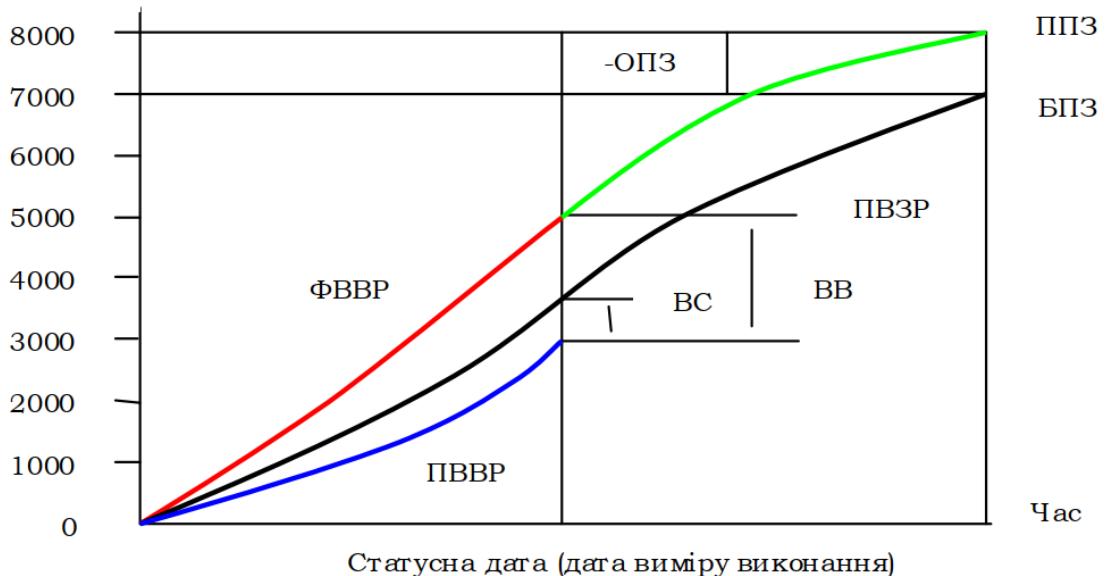


Рис. 9.3 Вартості робіт проєкту

Для полегшення обліку фактично виконаних робіт іноді використовують спрощені методики, які зводяться до дискретної оцінки частини виконаних робіт. Відповідно підраховують і планову вартість виконаних робіт.

Ці три параметри застосовують в різних комбінаціях для оцінки відповідності ходу виконання робіт запланованому. Найбільш часто використовують наведені нижче індикатори (рис. 9.3).

Відхилення за вартістю (BB) – індикатор відхилення фактичної вартості виконаних робіт від планової:

$$BB = ПВВР - ФВВР.$$

При правильному використанні він дозволяє оцінити вартісні відхилення і ймовірні тенденції як для окремих операцій, так і для груп операцій, фаз і проєкту в цілому.

Відхилення за вартістю у відсотках – відносний індикатор, що показує, яку частку від фактичної вартості виконаних робіт становить відхилення за вартістю:

$$BB\% = (BB / \Phi ВВР) \times 100.$$

Відхилення за строками (BC) - порівнює планову вартість виконаних робіт з бюджетом:

$$BC = ПВВР - ПВЗР.$$

Причому замість вартості, звичайно використовуваної як характеристика виконання, можна використати й інші визначальні параметри (обсяги робіт, людино-години, основні матеріали).

Програми управління проектами. Можливості програм управління проєктами в порівнянні й аналізі фактичних і планових вартостей і їх розподілу в часі, моделювання коригувальних впливів з урахуванням ресурсних і календарних обмежень, порівняння різних версій плану при аналізі «що коли» робить їх незамінними при аналізі складних проєктів.

Вихідна інформація контролю бюджету проєкту. Новий бюджет проєкту. Новий бюджет операцій проєкту, що залишилися, приймається до виконання, якщо ухвалено рішення про те, що застосування коригувальних впливів не є необхідним.

Запити на зміни. Рішення про необхідність коригування планового (базового) бюджету ініціює застосування процесів управління і часто приводить не тільки до зміни базового бюджету, але й планових строків виконання операцій, призначень ресурсів, технології й/або інших характеристик плану виконання проєкту.

Результати. Причини відхилень вартості виконання робіт повинні документуватися і включатися в архів проєкту й історичну базу даних для використання як у поточному проєкті, так і в інших проєктах виконуючої організації.

Питання до розділу 9

1. Процеси управління вартістю проєкту. Які складові?
2. Планування ресурсів проєкту. Складові вхідної інформації планування ресурсів?
3. Оцінка вартості операцій?
4. Розробка бюджету. Методики формування бюджету проєкту?
5. Аналіз ресурсів проєкту?
6. Контроль бюджету проєкту?

РОЗДІЛ 10. УПРАВЛІННЯ ЛЮДСЬКИМИ РЕСУРСАМИ

10.1. Процеси управління людськими ресурсами проекту

Людські ресурси проекту - це сукупність професійних, ділових, особистісних якостей учасників проекту та **членів команди проекту** та їх можливостей (теоретичних знань, здібностей, впливу, професіонального досвіду, компетентності, комунікаельності тощо), які можуть бути використані при здійсненні проекту.

Трудові ресурси – це певна частина людських ресурсів, що розглядаються як вимірюваній ресурс в проекті.

Персонал – це конкретні учасники проектної діяльності, індивідууми, особливістю яких є їх кваліфікація, щодо виконання функціонально-посадових обов'язків і ін., що описується в рамках штатного складу учасників проекту.

Управління людськими ресурсами проекту включає в себе процеси:

- організації;
- управління;
- керівництва **командою проекту**.

Процеси управління людськими ресурсами проекту:

– **Планування управління людськими ресурсами** – це процес визначення та документування ролей, відповідальності, необхідних навичок та підзвітності, а також створення плану управління людськими ресурсами.

– **Підбір та формування команди проекту** – це процес підтвердження доступності людських ресурсів та набору команди, необхідної для виконання завдань за проектом.

– **Розвиток команди проекту** – це процес підготовки та удосконалення компетенції, взаємодії членів команди проекту та покращення загальних умов роботи команди з метою підвищення ефективності виконання проекту.

– **Управління командою проекту** - це процес відстеження та оцінки діяльності членів команди, вирішення проблем та управління змінами, спрямований на оптимізацію виконання проекту.

Головна мета управління персоналом полягає в забезпеченні:

- такої поведінки кожного члена проектної команди, яка необхідна для **досягнення головних цілей проекту**, тобто – **успішної реалізації проекту загалом**;
- **створення команди проекту**, яка здатна якомога найбільш оптимально (за якістю, часом та витратами) реалізувати проект.

Основними сферами управління персоналом у проектах є:

- **лідерство** проектного менеджера;
- **розвиток** команди та **навичок** групової роботи;
- **мотивація** члені команда та команди у цілому;
- управління **конфліктами**.

Менеджер проекту – це керівник проекту, що обіймає постійну посаду в команді проекту й наділений повноваженнями в ділянці прийняття рішень щодо конкретних видів діяльності (організаційної, фінансової, кадрової, керівної та інших).

Лідерство як якість керівника проекту:

- це здатність справляти вплив на окремих індивідів і групи, спрямовуючи їхні зусилля на досягнення визначених та поставлених цілей;
- це здатність мобілізувати потенційні психологічні потреби послідовників (підлеглих) і спиратися на них в момент гострого суперництва чи конфлікту.

Якщо команда починається з лідера, то управління командою – з його знань і навичок організовувати роботу команди.

Ефективність роботи керівника визначається не його особистими рисами та манерою поведінки у стосунках з підлеглими.

Існує така класифікація стилів керівництва:

- **авторитарний** – керівник одноособово вирішує всі питання;
- **демократичний** – перед прийняттям рішень керівник радиться з колективом чи приймає колективне рішення;
- **ліберальний** – керівник чекає наказів від вищого керівництва, підкоряється рішенню своїх працівників.

Лідерство менеджера проекту виявляється у тому, що він дає завдання членам команди та наділяє їх повноваженнями у межах поставлених завдань з метою їх виконання.

Члени команди беруть на себе ці повноваження і відповідальність за виконання роботи.

Через наділення повноваженнями (**делегування**) менеджер проекту може:

- поліпшувати ефективність проектної команди;
- розвивати здібності працівників;
- сприяти зростанню компанії.

Делегування – це передача повноважень, відповідальності, контролю над важелями влади своєму підлеглуому.

Готовність керівника до делегування визначається наступними факторами:

- довіра до підлеглого;
- готовність передати владні повноваження;

- наявність ефективної системи контролю;
- здатність команди до делегування.

Делегування має три основних елементи:

- ***визначення функцій***, зобов'язань або завдань підлеглому;
- ***правильний розподіл повноважень***, щоб виконавець міг розпоряджатися необхідними для виконання завдання ресурсами;
- ***можливості та навички*** працівника щодо виконання завдання на належному рівні.

Менеджер проєкту як лідер повинен сприяти задоволенню:

- потреб завдань (визначати та досягати мети);
- потреб команди (будувати і координувати діяльність команди);
- індивідуальних потреб (задовольняти потреби членів команди).

10.2. Формування команди проєкту

Сучасна практика виконання проектів передбачає об'єднання виконавців проєкту в проєктні команди. Саме команда проєкту є найбільш гнучким елементом внутрішнього середовища проєктно-орієнтованої організації.

Згідно PMBOK (Project Management Body of Knowledge) управління персоналом проєкту включає в себе процеси організації команди проєкту та управління нею.

Поняття «команда проєкту» має чотири підходи до його визначення:

- «команда проєкту» як сукупність всіх осіб, залучених до виконання робіт по проєкту;
- «команда проєкту» як сукупність осіб, підзвітних керівнику проєкту та залучених до виконання робіт по проєкту;
- «команда проєкту» як будь-яка команда, що працює над виконанням проєкту;
- «команда проєкту» як група, що займається управлінням проєктом.

Команда проєкту – це група співробітників, що безпосередньо працюють над здійсненням проєкту й підлеглі керівникові; це група людей, що мають високу кваліфікацію в певній області й максимально відданих загальній цілі діяльності своєї організації, для досягнення якої вони діють спільно, взаємно погоджуючи свою роботу.

Команда проєкту складається з людей, кожному з яких призначена певна роль і відповідальність за виконання проєкту. Згідно методології управління командами

після розподілу ролей і відповідальності, кожний член команди має приймати участь у плануванні проєкту і прийняття рішень.

За формою команда проєкту відображає існуючу організаційну структуру управління проєктом, розділення функцій, обов'язків і відповідальності за рішення, що приймаються в процесі його реалізації. На верхньому рівні структури знаходиться менеджер проєкту, а на нижніх — виконавці, відділи і фахівці, що відповідають за окремі функціональні сфери.

За змістом команда проєкту є групою фахівців високої кваліфікації, що володіють знаннями і навичками, необхідними для ефективного досягнення цілей проєкту.

Виділяють наступні типи управлінських команд:

- традиційна;
- неформальні команди;
- формальна команда;
- проєктна команда.

Традиційна команда – це стабільний колектив людей, які знаходяться в безпосередньому підпорядкуванні керівника, який вирішує тактичні та стратегічні завдання структурного підрозділу.

Неформальна команда складається із співробітників різних підрозділів, які знаходяться на різних рівнях ієархії, які об'єдналися добровільно, і дозволяє вирішувати тактичні та стратегічні задачі, які стоять перед лідером.

Формальна група – це майбутня команда або невдала спроба формування управлінської команди. У формальній групі існують виражені ознаки проблемної команди, співробітники займають психологічні, а не управлінські ніші.

Проєктна команда складається із співробітників різних структурних підрозділів і підприємств (партнерів і замовників), які об'єднані в рамках проєкту.

Команда існує, поки не реалізований проєкт. При досягненні цілей проєкту його команда розформується. Таким чином, команда проєкту – це тимчасова, формально регламентована група спеціалістів, які створена для існування замислу проєкту і підпорядкована керівнику проєкту.

В організаційній структурі великих проєктів використовують три типи проєктних команд:

1. **Команда проєкту (КП)** – організаційна структура проєкту, яка створювана на період здійснення проєкту або однієї з фаз його життєвого циклу. Завданням керівництва команди проєкту є вироблення політики та затвердження стратегії

проекту для досягнення його цілей. У команду проекту входять особи, які представляють інтереси різних учасників проекту.

2. Команда управління проектом (КУП) – організаційна структура проекту, що включає тих членів КП, які безпосередньо залучені до управління проектом, у тому числі - представників деяких учасників проекту і технічний персонал. У відносно невеликих проектах КУП може включати в себе практично всіх членів КП. Завданням КУП є виконання всіх управлінських функцій і робіт у проекті по ходу його здійснення.

3. Команда менеджменту проекту (КМП) – організаційна структура проекту, яка очолювана керуючим (головним менеджером) проекту і створювана на період здійснення проекту або його життєвої фази. У команду менеджменту проекту входять фізичні особи, безпосередньо здійснюють менеджерські та інші функції управління проектом.

Головними завданнями команди менеджменту проекту є здійснення політики і стратегії проекту, реалізація стратегічних рішень і здійснення тактичного (ситуаційного) менеджменту.

В малих проектах обов'язки управління проектом можуть бути розподілені між усіма членами команди або доручені безпосередньо керівнику проекту.

Спонсор проекту працює в контакті з командою управління проектом і зазвичай бере участь у вирішенні таких питань, як фінансування проекту, прояснення змісту проекту та інших питань, що впливають на продуктивність і економічну ефективність проекту.

Задачею **менеджера проекту** при формуванні команди є підбір членів команди, які забезпечували б:

- відповідність кількісного і якісного складу команди цілям і вимогам проекту;
- ефективну групову роботу по управлінню проектом;
- психологічну сумісність членів команди;
- розгорнення ефективного і постійного спілкування відповідно до обраної структури процесів комунікацій у проекті;
- вироблення оптимальних групових розв'язань проблем, що виникають під час реалізації проекту.

Спонсор проекту призначає проект-менеджера, що здійснює загальне керівництво проектом, контролює його основні параметри і координує діяльність членів команди.

Менеджер проєкту визначає необхідну кількість фахівців членів команди, їх кваліфікацію, проводить відбір та укладає фінансові відносини з виконавцями проєкту.

10.3. Процеси управління командою проєкту

Процеси управління проектною командою включають в себе наступне:

1. **Планування необхідної кількості виконавців проєкту** – визначення та документальне оформлення ролей, відповідальності та підзвітності, а також створення плану управління людськими ресурсами.

2. **Набір команди проєкту** – залучення людських ресурсів, необхідних для виконання проєкту.

3. **Розвиток команди проєкту** – підвищення кваліфікації членів команди проєкту і зміцнення взаємодії між ними з метою підвищення ефективності виконання проєкту.

4. **Управління командою проєкту** – контроль над ефективністю членів команди проєкту, забезпечення зворотного зв'язку, рішення проблем і координація змін, спрямованих на підвищення ефективності виконання проєкту.

Система управління командою проєкту включає:

- організаційне планування;
- кадрове забезпечення проєкту;
- створення команди проєкту;
- функції контролю і мотивації трудових ресурсів проєкту для ефективного ходу робіт і завершення проєкту.

Система управління командою проєкту націлена на керівництво і координацію діяльності команди проєкту, використовує стилі керівництва, методи мотивації, адміністративні методи, підвищення кваліфікації кадрів на всіх фазах життєвого циклу проєкту.

Керівництво проєкту:

- спостерігає за діяльністю команди;
- залагоджує конфлікти, вирішує проблеми;
- дає оцінку ефективності роботи членів команди.

Результатами управління командою проєкту є **запити на зміну**, поновлення плану управління людськими ресурсами, вирішення проблем, надання вхідної інформації для оцінки ефективності роботи і накопичення знань персоналу організації.

Менеджери проектів повинні вміти визначати, формувати, підтримувати, мотивувати, керувати і надихати команди проектів для підвищення ефективності їх роботи та досягнення цілей проекту.

10.4. Мотивація роботи команди проекту

Мотивація — це стимулювання людини чи групи людей до активізації діяльності для досягнення цілей організації (проекту); це сукупність сил, які спонукають людину займатися діяльністю з витратою певних зусиль на певному рівні старання й сумлінності з певним ступенем наполегливості в напрямку досягнення певних цілей.

Як правило, виконавці з високим рівнем мотивації мають значну продуктивність праці і отримують задоволення від роботи, від досягнення поставленої перед групою мети.

Існує багато чинників, які можуть мотивувати людей.

Менеджер проекту не є таким чинником, але його завдання зрозуміти напрямки їх зацікавленості та можливості мотивації членів команди проекту і забезпечувати реалізацію мотивації, як моральної так і фінансової.

Програма мотивації – це система заходів, виконуваних протягом певного проміжку часу, які спрямовані на стимулювання певних співробітників з метою отримання певних результатів у підвищенні якості та ефективності результатів їх роботи.

Програма мотивації включає:

- цілі (до чого необхідно стимулювати співробітників);
- охоплення (категорії співробітників і проектів, до яких вона застосовується);
 - термін дії (наприклад, півроку або рік) критерії, процедури оцінки й відповідальних за оцінку поведінки різних категорій співробітників;
 - систему заохочень і стягнень;
 - календарний план заходів та відповідальних за їх виконання;
 - бюджет програми мотивації.

Найчастіше метою програми мотивації є **підвищення ефективності при збереженні необхідного рівня якості**. При цьому для кожної категорії співробітників ефективність і якість визначаються і оцінюються по-різному.

Основний принцип мотивації полягає в тому, що **заохочення або стягнення** повинні накладатися на співробітника тільки за результати робіт, **доручених**

безпосередньо йому. Тому керівники проекту і функціональні менеджери повинні преміюватися за виконання проекту в цілому.

Найбільш часто використовувані механізми матеріального стимулювання передбачають **розрахунок премії (чи бонусів)**, виходячи з таких показників:

- прибуток (різниця між виторгом за проектом й собівартістю, розрахованою методом повного розподілу витрат) або маржинальний прибуток (різниця між виторгом за проектом і собівартістю, розрахованою за змінними витратами);
- економія витрат.

10.5. Управління конфліктами в проектах

Конфлікт – це відсутність згоди між двома чи декількома суб'єктами, зіткнення протилежних сторін, сил, які можуть бути конкретним особами або групами працівників, а також внутрішній дискомфорт однієї людини.

В умовах проекту конфлікти неминучі і досить важливі, як з точки зору їх причин, так і з точки зору можливих впливів на роботу проектної команди та результати виконання проекту.

Джерелами конфліктів можуть стати дефіцит ресурсів, розстановка пріоритетів при складанні розкладу або особисті стилі роботи. Наявність прийнятих в команді принципів, норм і усталеної практики управління проектами, наприклад планування комунікацій і визначення ролей, сприяє зниженню кількості виникаючих конфліктів.

Успішне врегулювання конфліктів призводить до більш високої продуктивності та позитивним робочим взаємовідносинам.

Конфлікт може бути позитивним, якщо він:

- є основою для початку дискусії з обговорення того чи іншого питання;
- розв'язанні того чи іншого питання;
- покращує стосунки між людьми;
- дає змогу зняти напруженість;
- дає змогу працівникам повніше розкрити свої можливості.

Конфлікт може бути негативним якщо він:

- відриває людей від розв'язання важливих питань;
- викликає почуття невдоволеності в колективі;
- веде до особистісної або групової ізоляції, а також протидіє порозумінню.

Класична точка зору на конфлікт полягала в тому, що він не повинен виникати.

Але визнано, що певна ступінь конфліктності обов'язкова у відносинах.

Основними причинами конфліктів у проєкті є:

- пріоритеті в проєкті;
- вартість проєкту;
- терміни, календарний план;
- адміністративні процедури;
- ресурси;
- погляди на реалізацію проєкту;
- особистості.

Протягом часу реалізації проєкту виникають різні джерела конфліктів.

Стадії конфлікту:

I. Виникнення об'єктивної конфліктної ситуації.

II. Усвідомлення конфлікту.

III. Конфліктні дії.

IV. Зняття або вирішення конфлікту.

Функції конфлікту:

- конструктивна (джерело ідей, поява нових напрямків) - спільний пошук вирішення конфлікту з вигодою для обох сторін.

- деструктивна - учасники залишаються при свій думці.

Можна визначити наступні розповсюджені методи управління конфліктом.

Метод ухилення. Він базується на тому, що людина намагається відійти від конфлікту, уникнути ситуації, що провокує протиріччя та уникнути обговорення питання, що приводить до конфлікту.

Метод пристосування. Цей стиль характерний при природному небажанні уникнути конфлікта, тобто необхідно стимулювати почуття спільноті в колективі.

Метод компромісу. Він характеризується прийняттям точки зору іншої сторони, але до певної межі. Проєкт-менеджер може ефективно його використовувати при офіційних переговорах по контракту і при неформальних переговорах з учасниками проєкту.

Метод форсування. Примус до прийняття однієї точки зору. Цей стиль ефективний, коли керівник має велику владу над підлеглими.

Метод вирішення проблем. Це визнання розбіжностей у думках і готовність ознайомитись з іншими точками зору, щоб краще зрозуміти причину конфлікту та знайти вихід прийнятний для всіх. Вирішення проблеми є синтезом всіх методів управління конфліктами і використовується, коли є досить часу і існує довіра між конфліктними сторонами.

Вирішення конфліктів — це усунення повністю або частково причин, які провокують конфліктну ситуації.

Виявлення і розв'язання конфліктів вимагає від менеджера проекту виконання таких п'яти задач:

- 1) *передбачення потенційних конфліктів* і, по можливості, здійснення превентивних дій;
- 2) *отримання інформації* про конфлікти по мірі їх виникнення і пошук розуміння *їх основних причин*;
- 3) переконання членів команди на першому етапі *спробувати розв'язати конфлікти один з одним* чи всередині свого функціонального підрозділу;
- 4) спроба *прийняти компромісне рішення* для того, щоб обидві сторони досягли своїх цілей;
- 5) пошук допомоги *з боку спонсора* проекту в розв'язанні конфліктів, які знаходяться поза компетенцією менеджера проекту.

Питання до розділу 10

1. Які особливості управління людськими ресурсами в проекті?
2. Які види команд проєкту ви знаєте. Охарактеризуйте?
3. Процеси управління командою проєкту?
4. Мотивація роботи команди проєкту?
5. Джерела та властивості конфліктів у командах?
6. Методи управління конфліктами?

РОЗДІЛ 11. КОНТРОЛЬ ХОДУ ВИКОНАННЯ ПРОЕКТУ

11.1. Контроль проектної діяльності

Контроль проектної діяльності — це процес, у якому керівник проєкту встановлює, чи досягнуто поставлені цілі, виявляє причини дестабілізації процесу виконання роботи і обґруntовує прийняття управлінських рішень, що коригують виконання завдань, раніше, ніж буде нанесено збиток виконанню проєкту (зрив строків виконання робіт, перевищення використання ресурсів і вартості, низька якість тощо).

Метою процесу контролю проєкту, а точніше, процесу «Відстежування ходу проєкту і контроль за ним» (Project Monitoring and Control) є надання інформації, необхідної для розуміння ходу проєкту, для того, щоб дозволити керівництву виконувати управляючі дії в ситуаціях, коли хід проєкту істотно відрізняється від запланованого.

Контролем виконання проєкту називається регулярний вимір параметрів проєкту та ідентифікація виникаючих відхилень.

11.2. Контроль виконання проєкту

Регулярний вимір параметрів проєкту та ідентифікація виникаючих відхилень називається *контролем виконання проєкту*.

Можна виділити дві складові *контролю виконання проєкту*:

- організація і контроль виконання проєкту;
- аналіз і регулювання виконання проєкту.

Організація і контроль виконання проєкту (Project Performance) – це організація виконання включених до плану проєкту робіт і контроль їх виконання.

Аналіз і регулювання виконання проєкту (Project Controlling) – це процес порівняння фактичного виконання із запланованим, аналіз відхилень, оцінка можливих альтернатив і прийняття, в разі необхідності, коригуючих дій для ліквідації небажаних відхилень від базового рівня показників.

Організація і контроль, аналіз і регулювання являють собою досить складні управлінські процеси. Крім того, це найбільш витратна частина реалізації проєкту, адже саме на цих етапах і створюється сам продукт. Дані етапи характеризуються підвищеним споживанням ресурсів (приблизно 80% всіх ресурсів).

Організація і контроль виконання проєкту зазвичай розглядається як організація виконання включених до плану проєкту робіт і контроль їх виконання.

Організація і контроль виконання проєкту включає:

- організацію управління предметною областю проєкту;
- контроль виконання проєкту по часових параметрів;
- організацію і контроль виконання проєкту за вартістю;
- організацію і здійснення контролю якості;
- оперативне управління заходами по зниженню ризиків;
- вдосконалення команди проєкту;
- розподіл інформації;
- організацію і підготовку контрактів в проєкті;
- організацію управління змінами в проєкті.

Основною метою контролю проєкту є забезпечення виконання планових показників і підвищення загальної ефективності функцій планування і контролю проєкту.

Зміст контролю проєкту полягає у визначенні результатів діяльності на основі оцінки і документування фактичних показників виконання робіт і порівняння їх з плановими показниками.

Принципи побудови ефективної системи контролю:

- наявність конкретних планів;
- наявність інформативної системи звітності;
- наявність ефективної системи аналізу фактичних показників і тенденцій;
- наявність ефективної системи реагування;

Об'єкти контролю:

- зміни змісту;
- розклад;
- витрати;
- якість;
- ризики.

Аналіз і регулювання виконання проєкту – це процес порівняння фактичного виконання із запланованим (рис. 11.1), аналіз відхилень, оцінка можливих альтернатив і прийняття, в разі необхідності, коригуючих дій для ліквідації небажаних відхилень від базового рівня показників.

Аналіз і регулювання виконання проєкту включає:

- аналіз стану і регулювання предметної області проєкту;
- аналіз і регулювання проєкту по часових параметрах;
- аналіз і регулювання проєкту за вартісними показниками;

- аналіз стану і забезпечення якості;
- аналіз ризиків;
- аналіз діяльності та розвиток команди проєкту;
- аналіз комунікацій при виконанні проєкту;
- контроль і регулювання контрактів;
- аналіз, інтеграція і регулювання змін в проєкті.

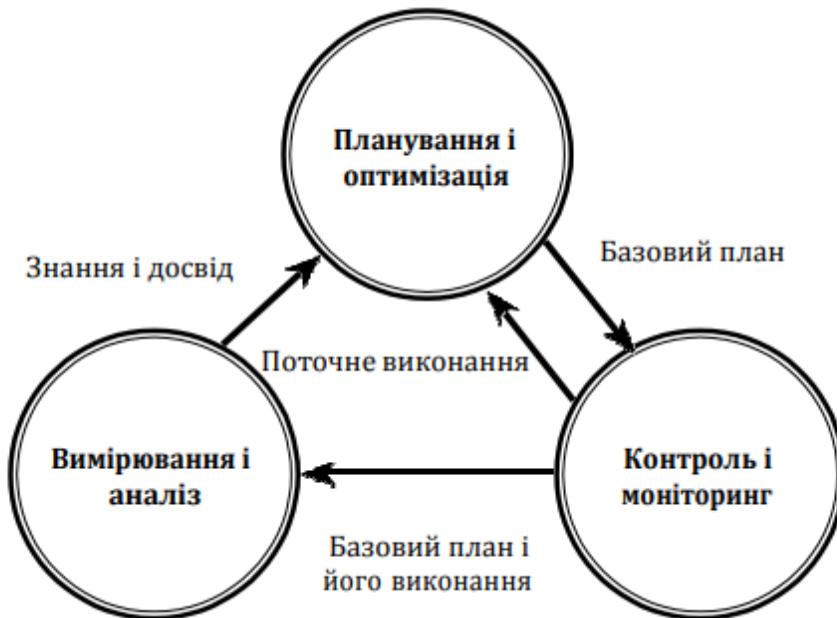


Рис. 11.1 Взаємозв'язок процесів планування, контролю і аналізу

11.3. Способи контролю ходу проєкту

- Порівняння фактичних показників** (обсягу виконаних робіт, витрачених зусиль, засобів) з **плановими**.
- Контроль прийняття і виконання рішення про зміну плану.**
- Виконання дій, що коригують план:**
 - перегляд поточного плану і внесення змін до нього;
 - виконання робіт по пом'якшенню дії ризиків, що відбулися;
 - припинення виконання проєкту і визначення нових цілей, взяття нових зобов'язань.
- Планування процесу контролю** виконання проєкту.
- Забезпечення процесу контролю відповідними ресурсами.**
- Призначення персональної відповідальності і повноважень.**

- g. **Навчання персоналу, який виконуватиме моніторинг.**
- h. **Розробка форматів документів процесу.**
- i. **Виконання процесу** – власне контроль за ходом проєкту.
- j. **Відстеження процесу** на предмет відповідності прийнятым стандартам.
- k. **Обговорення результатів процесу з вищим керівництвом.**

Вимоги до системи контролю – точність, своєчасність, повнота інформації, забезпечення єдності інформації для всіх учасників проєкту.

Мета й призначення контролю – визначаються необхідністю максимальної відповідності поточних значень параметрів виконання проєкту запланованим. На процес реалізації проєкту впливає багато як зовнішніх, так і внутрішніх дестабілізуючих факторів та умов. Це призводить до зміни запланованих параметрів проєкту (строкових і вартісних).

Предмет контролю – це факти і події, перевірка виконання конкретних рішень, з'ясування причин відхилення, оцінка ситуації, прогнозування наслідків. Контроль передбачає постійне спостереження за ходом реалізації проєкту.

Об'єкти контролю – це час, вартість, якість, зміни, які виникають у ході реалізації проєкту; підготовка, отримання, розподіл і схвалення документів проєкту, стан справ із фінансуванням, експлуатаційні характеристики проєкту, відповідність положенням контракту тощо.

11.4. Процеси контролю

До **процесів контролю** включають:

- визначення результатів діяльності на основі зіставлення результатів здійснення рішень із запланованими;
- порівняння показників очікуваного й фактичного виконання планів;
- аналіз ймовірних відхилень від запланованих показників;
- перевірка припущень;
- перевірка методичної та змістової узгодженості планового процесу;
- проведення необхідних робіт для виправлення ситуації.

Основні процеси контролю проєкту: загальний контроль змін, ведення звітності проєкту.

Допоміжні процеси контролю проєкту: процеси контролю розкладу, витрат, якості, ризику, змін змісту.



Рис. 11.2 Система контролю проекту

Система контролю – комплексу формалізованих, документованих методик, використовуваних в рамках проекту для визначення інформації, що збирається, методів її аналізу, способів реагування на відхилення і відповідальних осіб (рис. 11.2).

11.5. Методи та види контролю проекту

У системі контролю проекту такі **методи контролю**:

- **контроль проекту по часовим параметрам** (дорожня карта проекту), чи план проекту по вікам;
- **контроль проекту по вартісним параметрам.**

Існують **три основні види контролю**:

- попередній;
- поточний;
- заключний.

Попередній контроль здійснюється до фактичного початку виконання робіт і направлений на дотримання певних правил і процедур, як правило, він торкається ресурсного забезпечення робіт.

Поточний контроль здійснюється при реалізації проекту, він включає:

- контроль часу;
- досягнення проміжних цілей проекту;
- виконання заданих обсягів робіт;
- контроль бюджету;
- контроль ресурсів;
- контроль якості.

Основна мета поточного контролю – оперативне регулювання ходу виконання проєкту відповідно до паспорту проєкту та плану проєкту.

В залежності від необхідної точності розрізняють такі **технології поточного контролю**:

- контроль на момент закінчення робіт;
- контроль на момент 50% готовності робіт;
- контроль у заздалегідь установлених певних точках проєкту;
- регулярний оперативний контроль;
- експертна оцінка ступеня виконання робіт і готовності проєкту.

Заключний контроль проводиться на стадії завершення проєкту з метою інтегральної оцінки реалізації проєкту.

Основним призначенням його є **узагальнення отриманого досвіду** для подальшої розробки й реалізації аналогічних проєктів з метою вдосконалення процедур управління.

Дуже часто помилково визначають оцінку проектної діяльності як контроль проектної діяльності. Як **контроль**, так і **оцінка проектної діяльності** є дуже **важливою функцією зворотного зв'язку** при аналізі проектної діяльності для визначення розбіжностей у виконанні проєкту.

Але між контролем та оцінкою існує **багато істотних розбіжностей**:

- контроль передбачає **постійне спостереження** за просуванням проєкту, а оцінка базується на **періодичному визначенні** проміжних підсумків;
- **контроль** проектної діяльності **сфокусований на деталях** того, що відбувається у проєкті, а **оцінка** сфокусована на інтегральній загальній картині;
- за **контроль відповідає керівник проєкту**, а оцінку здійснює особа чи група осіб, які не працюють безпосередньо над проєктом – зовнішні зацікавлені особи проєкту, представники спонсора проєкту чи спеціалісти аналітики.

Відповідно до розбіжностей можна дати таке визначення **оцінки проектної діяльності** – це об'єктивне періодичне підбиття підсумків для визначення статусу проєкту щодо реалізації його сформульованих цілей. Оцінку здійснюють під час реалізації проєкту та після його завершення.

11.6. Моніторинг

В ході відстеження стану виконання проєкту керівнику проєкту потрібно вміти визначати, чи вкладається проєкт в запланований бюджет і чи буде він завершений в заплановані терміни. Для цього мало повсякденно збирати фактичні дані про хід робіт,

ще потрібно й правильно їх аналізувати. Для цього використовуються методи моніторингу.

Моніторинг – це оперативний контроль, стеження, облік, аналіз і складання звітів про фактичне виконання проекту в порівнянні з планом.

Методи моніторингу:

- метод простого контролю (0% і 100%);
- метод детального контролю;
- метод 50/50 (ступінь завершеності роботи визначається в момент, коли на роботу витрачено 50% бюджету);
- метод по віках.

1. Метод простого контролю – метод, що відслідковує тільки моменти завершення детальних робіт, при якому існують лише дві міри завершеності роботи: 0% і 100% (метод «0—100»).

Іншими словами, вважається, що робота виконана лише тоді, коли досягнутий її кінцевий результат. Можна відзначити, що метод простого контролю придатний до великої кількості короткочасних робіт, які з'явилися завдяки деталізації робіт. В рамках короткочасних робіт не потрібно визначення їх проміжних станів. Крім того, визначити проміжний стан роботи складно.

2. Метод детального контролю, який передбачає виконання оцінок проміжних станів виконання роботи (наприклад, завершеність детальної роботи на 50% означає, що, по оцінках виконавців і керівництва, цілі роботи досягнуті наполовину). Даний метод використовують, якщо тривалість роботи велика і цей метод складніший, оскільки вимагає від менеджера оцінювати відсоток завершеності для робіт, що знаходяться в процесі виконання.

3. Метод 50/50, в якому є можливість обліку деякого проміжного результату для незавершених робіт. Міра завершеності роботи визначається в мить, коли на роботу витрачено 50% бюджету.

4. Метод по віках. Застосовується для тривалих проектів із значної кількістю різноманітних типів робіт, які виконуються різними виконавцями чи групами виконавців та співвиконавців проекту.

Цей метод полягає у **визначенні достатньої кількості проміжних результатів роботи і контролю їх** по простому, дискретного методу: досягнутий чи ні. Однак в рамках роботи кожен результат може мати власну «вагу» – досягнення чергового результату інтерпретується як певний відсоток загального виконання роботи.

Специфіка методу полягає:

- робота ділиться на частини відмінами;
- кожна з яких має на увазі певну міру завершеності роботи;
- встановлюємо віхи перед початком роботи;
- Фіксуємо досягнення віх на звітну дату.

В ході моніторингу проєкту відбувається *порівняння ходу проєкту з планом*.

Для цього необхідно виконати такі види робіт:

1. *Моніторинг ключових показників проєкту*. Це моментальний «знімок» атрибутів створюваних продуктів.

2. *Моніторинг зобов'язань за проєктом*. Виявлення виконаних зобов'язань (як зовнішніх, так і внутрішніх), невиконаних зобов'язань або тих зобов'язань, які можуть бути не виконані через появу певних ризиків.

3. *Моніторинг ризиків проєкту*. Виявлення в контексті поточного ходу виконання проєкту переліку ризиків зі всіма їх характеристиками: вірогідністю виникнення, ступенем дії тощо.

4. *Аналіз та оцінка прогресу проєкту*. *Прогрес проєкту* — це просування виконання робіт проєкту в напрямку досягнення його цілей.

5. *Аналіз контрольних точок проєкту*. Це формальна процедура, що виконується по досягненню певної віхи (*milestone*). Обговорюються всі аспекти виконання проєкту, виконується ретельне вивчення поточної ситуації. Здійснюється *аналіз тенденцій віх* — це простий метод для аналізу по датам результатів робіт у проєкті у порівнянні їх із плановими даними. Результати аналізу контрольних точок документуються.

Виділяють 5 можливих варіантів дій у випадку відхилень проєкту від плану:

1. **Знаходження альтернативного рішення**. У першу чергу необхідно розглянути можливості, пов'язані з підвищенням ефективності робіт за рахунок нових технологічних або організаційних рішень. Нове рішення, наприклад, може полягати у зміні послідовності виконання ряду робіт;

2. **Перегляд вартості**. Даний підхід означає збільшення обсягів робіт і призначення додаткових ресурсів. Рішення може полягати у збільшенні навантаження на існуючі ресурси або залученні додаткових людей, обладнання, матеріалів. Даний підхід зазвичай застосовується у разі необхідності усунення тимчасових затримок проєкту.

3. **Перегляд термінів**. Даний підхід означає, що терміни виконання робіт будуть відсунуті. Керівництво проєкту може піти на таке рішення в разі жорстких обмежень по вартості.

4. Перегляд змісту робіт. Даний підхід передбачає, що обсяг робіт за проєктом може бути зменшений і відповідно лише частина запланованих результатів проєкту буде досягнута. Відзначимо, що мова не йде про перегляд якісних характеристик одержуваних результатів проєкту.

5. Припинення проєкту. Це, мабуть, найбільш складне рішення. Однак воно має бути прийняте, якщо прогнозовані витрати за проєктом перевищують очікувані вигоди. Рішення, пов'язане з припиненням проєкту, крім суто економічних аспектів, пов'язано з подоланням проблем психологічного характеру, пов'язаних з інтересами різних учасників проєкту.

Питання до розділу 11

1. Які є стадії контролю виконання проєкту?
2. Що включає організація і контроль виконання проєкту?
3. Які є об'єкти контролю проєкту?
4. Що таке моніторинг проєкту?
5. Що включає аналіз і регулювання виконання проєкту?
6. Методи моніторингу проєкту?

РОЗДІЛ 12. УПРАВЛІННЯ ЗМІНАМИ, ПОСТАВКАМИ ТА ЯКІСТЮ ПРОЄКТУ

12.1. Управління змінами в процесі виконання проєкту

Управління змінами в проєкті (Project Change Management) – це розділ управління проєктами, що включає в себе формальні процеси і процедури для інтеграції і управління змінами в проєкті, здійснюваними протягом його життєвого циклу.

Управління змінами в проєкті складається з:

- прогнозування;
- планування;
- здійснення;
- контролю;
- регулювання змін.

Зміни в обсягах проєкту — чи не одна з найголовніших причин зростання вартості проєкту і збільшення часу його виконання. Дуже часто ці зміни підвищують витрати на 50% і більше. Тому однією з найважливіших функцій менеджера проєкту є контроль за змінами у проєкті.

Під зміною розуміється заміщення одного рішення іншим внаслідок впливу різних зовнішніх і внутрішніх факторів при розробці та реалізації проєкту. Зміни можуть вноситися в різні розділи проєкту.

Ці зміни впливають на виконання проєкту таким чином:

- підвищують затрати;
- спричиняються до затримки виконання проєкту;
- знижують продуктивність праці виконавців робіт;
- погіршують стосунки між членами команди.

Причинами внесення змін зазвичай є неможливість передбачення на стадії розробки проєкту нових проєктних рішень, більш ефективних матеріалів, конструкцій і технологій і т. ін., а також відставання в ході реалізації проєкту від запланованих термінів, обсягів внаслідок непередбачених обставин.

Причинами змін в змісті робіт можуть бути:

1. Зміни кон'юнктури на ринку.
2. Дії і наміри конкурентів.
3. Технологічні зміни, зміни в цінах і доступності ресурсів.
4. Економічна нестабільність.

5. Помилки в планах і оцінках.
6. Помилки у виборі методів, інструментів, організаційної структури або стандартів.
7. Зміни в контрактах і специфікаціях.
8. Затримки поставок або поставки, які не відповідають вимогам якості.
9. Необхідність прискорення робіт.
10. Вплив інших проєктів.

Управління змінами тісно пов'язане з усіма процесами і функціями в проєкті, розглянутими раніше. При своєму здійсненні проєкт може піддаватися різним змінам (змінам факторів):

- у предметній області та її конфігурації (*Configuration Management*);
- у часі;
- за вартістю;
- за якістю;
- за ризиками;
- за контрактами;
- за поставками;
- за людськими ресурсами;
- за комунікаціями;
- у процесах управління проєктом на всіх фазах його життєвого циклу.

Особливості управління змінами

Управління змінами в проєкті необхідно розглядати як всеосяжний інтегральний процес, що має відношення до:

- всіх внутрішніх і зовнішніх впливів на проєкт;
- прогнозування можливих змін в проєкті;
- визначення змін, що вже трапилися;
- планування дій, попереджуючих негативні впливи на проєкт;
- управління в проєкті прийнятими змінами;
- координації змін по всьому проєкту.

Управління змінами покликане забезпечити вирішення проблем і завдань, пов'язаних із забезпеченням захисту проєкту від можливого негативного впливу зовнішніх і внутрішніх факторів, внесенням необхідних скоординованих змін і контролем за їх ефективним здійсненням.

Система контролю за змінами вирішує такі завдання:

- визначає зміни відносно початкового обсягу;

- прогнозує витрати, час і вплив цих змін на інші роботи;
- фіксує інформацію щодо їх запровадження;
- інформує про них зацікавлених осіб;
- запроваджує систему вирішення суперечностей з мінімальними конфліктами.

Систему контролю за змінами інколи називають «прогнозуванням трендів», «контролем відхилень», «контролем за формами».

Дуже важливо запровадити її якомога раніше. За цією системою готуються тижневі або місячні огляди на стадіях розробки ІТ-систем і постачань програмних та технічних засобів. Контроль здійснюється за допомогою оперативного звітування щодо змін та обговорювання їх необхідності і наслідків (стосовно затрат і часу) у колі провідних спеціалістів.

Процес управління змінами в проекті включає:

1. Розробку концепції управління змінами в проекті:

- вироблення стратегії управління змінами;
- аналіз можливих змін;
- визначення принципів інтеграції процесів управління змінами;
- затвердження концепції.

2. Прогнозування і планування змін:

- вибір методів і засобів прогнозування і планування змін;
- прогнозування змін;
- моніторинг зовнішнього середовища і тенденцій змін;
- планування можливих попереджувальних впливів для захисту проекту;
- розробка плану управління змінами в проекті.

3. Організацію та контроль змін в проекті:

- розподіл ролей і відповідальності персоналу, залученого в управління змінами, і формування відповідної організаційної структури;
- твердження процедур здійснення змін в проекті;
- введення в дію системи управління змінами;
- інформаційна підтримка управління змінами в проекті;
- збір та аналіз запитів і пропозицій на внесення змін;
- прийняття рішень і внесення змін в проект;
- ведення бази даних змін проекту.

4. Аналіз і регулювання змін:

- контроль здійснення змін в проекті;

- огляд і аналіз динаміки змін в проекті;
- поточна оцінка змін в проекті і досягнутих в зв'язку з цим результатів;
- звіт про виконання змін в проекті і відхиленнях від плану управління змінами;
- пропозиції щодо коригування плану змін.

5. Завершення управління змінами в проекті:

- після проектний аналіз, оцінка змін і їх результатів;
- заключний звіт про фактичні зміни в проекті;
- формування архіву змін в проекті;
- коригування стратегії на майбутнє і формування бази знань управління змінами.

Управління змінами в ряді випадків є частиною або наслідком необхідності проведення дій, пов'язаних з управлінням конфліктами в проектній діяльності та реалізує процеси щодо усунення або зменшення конфліктних ситуацій.

12.2. Управління поставками

Управління поставками і контрактами в проекті (Project Contracts Management) – це розділ управління проектами, що включає процеси, необхідні для забезпечення постачання продуктів і послуг ззовні. Включає в себе планування поставок і послуг, планування пропозицій, запит, вибір джерел, адміністрування контракту, закриття контракту. Управління контрактами і поставками в проекті включає процеси, спрямовані на придбання ззовні продуктів і послуг, необхідних для виконання проекту.

Управління контрактами і поставками в проекті включає:

1. Розробку концепції управління контрактами в проекті:

- проведення маркетингу ринку продуктів і послуг;
- розробка стратегії управління контрактами;
- складання специфікації продуктів і послуг;
- визначення можливих джерел придбання ресурсів.

2. Планування поставок і контрактів для забезпечення необхідних продуктів і послуг:

- визначення потреби проекту в продуктах і послугах;
- проведення маркетингових досліджень для визначення можливих постачальників і виконавців;
- вибір методу забезпечення і підтримки контрактів в проекті;

- визначення типів контрактів;
- визначення титульного списку робіт і переліку контрактів в проєкті;
- формування графіка укладення контрактів.

3. Організацію та підготовку контрактів в проєкті:

- розподіл функціональних обов'язків та відповідальності відповідно до плану управління контрактами;
- підготовка документації, необхідної для проведення тендера;
- запрошення на тендери торги;
- проведення торгів і вибір претендентів;
- заключення контрактів;
- розробка системи звітності та порядку внесення змін.

4. Контроль і регулювання контрактів:

- організація системи контролю контрактів;
- облік виконання робіт за контрактом;
- визначення стану і прогноз виконання робіт і їх забезпечення;
- подання звітності про виконання контрактів;
- аналіз поточного стану виконання контрактів та запитів на зміни;
- вирішення спорів і розбіжностей.

5. Завершення управління контрактами в проєкті:

- приймання результатів виконання контрактів;
- заключний аналіз і оцінка ефективності забезпечення проєкту;
- закриття контрактів;
- заключний звіт по управлінню контрактами в проєкті;
- формування архіву контрактної документації;
- формування шаблонів контрактів і бази знань з управління поставками.

12.3. Управління якістю у проєкті

Управління якістю в проєкті (*Project Quality Management*) – розділ управління проєктами, що включає в себе процеси, необхідні для забезпечення гарантій того, що проєкт задовольнить потребам, заради яких він і був зроблений. Включає планування якості, забезпечення якості і контроль якості. Управління якістю здійснюється протягом усього часу виконання проєкту:

- проєктні, організаційні та управлінські рішення;
- використовувані матеріали, обладнання сировину і ін.;

- якість виконуваних робіт при реалізації проєкту;
- якість отриманих результатів проєкту (продукти проєкту, надані послуги).

Створення і підтримка якості процесів і продуктів в ході виконання проєкту вимагає систематичного підходу. Такий підхід повинен гарантувати, що:

- потреби замовника зрозумілі і задоволені;
- враховані потреби інших учасників проєкту;
- врахована методика розробника в області управління якістю проєкту.

Слово «якість» часто вживають для позначення елітарності, високої вартості, відповідності найвимогливішим побажанням споживачів.

Міжнародний стандарт ISO 8402 визначає **якість** як сукупність властивостей і характеристик об'єкта, що гарантують його можливість задоволити явні та неявні потреби споживачів.

Якість – це «ступінь, в якій сукупність внутрішніх характеристик відповідає вимогам».

Низька якість – це завжди проблема. Менеджер проєкту та команда управління проєктом відповідають за визначення та забезпечення необхідних рівнів як якості.

Якість проєкту — це ступінь відповідності всіх його характеристик вимогам проєкту.

Як головний параметр якості проєкту постає якість продукту (послуги), що є результатом виконання проєкту.

Якість продукту проєкту означає відповідність вимогам споживача (цілям замовника).

Ключові аспекти якості:

1. Якість продукту проєкту як відповідність ринковим потребам і сподіванням споживачів.
2. Якість розробки і планування проєкту.
3. Якість виконання робіт за проєктом відповідно до планової документації.
4. Якість ресурсів, що залучаються до виконання проєкту.
5. Якість експлуатації продукції проєкту.
6. Якість розвитку продукції проєкту.
7. Якість утилізації та переробки продукту після використання.

Менеджмент якості проєкту

Сучасний менеджмент якості проєкту базується на таких основних принципах:

- якість – це не самостійна функція управління, а невід'ємний елемент проєкту;

- якість – це те, чого очікує споживач;
- відповіальність за якість проєкту має бути адресною;
- підвищувати якість можна лише зусиллями всіх працівників;
- контролювати завжди ефективніше процес, аніж результат;
- програма забезпечення якості мають бути частиною загального плану проєкту.

Стандарт є основним нормативно-технічним документом, в якому показники якості встановлюються, виходячи з новітніх досягнень науки, техніки і попиту споживачів.

Стандарти в управлінні якістю проєкту – ISO 100006 «Системи менеджменту якості. Настанови з управління якістю в проєктах»

У стандарті ISO 100006 процеси згруповані у дві категорії:

- процеси управління проєктом,
- процеси, пов’язані з продуктом проєкту (проєктування, виробництво, перевірка)

План управління якістю – це документ, у якому регламентовано **конкретні заходи у сфері якості, ресурси та їх послідовність** щодо конкретної продукції, проєкту чи контракту:

- контроль якості розробки проєкту;
- контроль документування проєкту;
- контроль поставок та роботи субпідрядників;
- контроль якості розробки;
- контроль випробувань (тестування) програмних засобів;
- коригувальні дії (план і процедура).

В такому плані має бути описана:

Система якості — це сукупність організаційної структури, методик, процесів і ресурсів, необхідних для управління якістю. Вона призначена для задоволення внутрішніх потреб організації.

Настанова з якості — це документ, в якому викладено політику у сфері якості і описано систему якості організації. Настанова з якості може охоплювати всю діяльність організації чи тільки її частину.

Програма якості — документ, який регламентує конкретні заходи у сфері якості, ресурси і послідовність діяльності, що відноситься до специфічної продукції, проєкту чи контракту.

Забезпечення якості – це регулярна перевірка ходу реалізації проєкту з метою встановлення відповідності визначеному раніше вимогам до якості; це система послідовних запланованих і реалізованих робіт для підтвердження того, що проєкт задовольняє відповідні стандарти; оцінка загального виконання проєкту на регулярній основі для підтвердження того, що проєкт задовольняє стандарти якості.

Призначення процесу — впровадження запланованих, систематичних операцій, які забезпечують використання в проєкті всіх процесів, необхідних для виконання вимог з якості.

Здійснення забезпечення якості – це один із процесів виконання, в якому використовуються дані, отримані під час здійснення контролю якості.

Забезпечення якості продукту вимагає:

1. Чіткої специфіки: точно визначити кінцевий продукт, проміжні результати проєкту.
2. Використання прийнятих стандартів.
3. Попереднього практичного досвіду.
4. Кваліфікованого персоналу.

Контроль якості — це:

- відслідковування певних результатів по проєкту для встановлення того, чи відповідають вони показникам та стандартам якості, і для визначення шляхів усунення причин нездовільного виконання;
- відслідковування конкретних результатів діяльності по проєкту в цілях визначення їх відповідності стандартам і вимогам з якості і визначення шляхів усунення причин невідповідності;
- процес контролю і запису результатів дій, спрямованих на забезпечення якості, для оцінки виконання й розробки рекомендацій щодо необхідних змін.

Особливості контролю якості:

- Контроль якості здійснюється протягом всього проєкту.
- До результатів проєкту відносяться як результати робіт, так і управлінські результати, такі як показники виконання вартості та строків.
- Контроль якості проєкту покликаний забезпечити усунення будь-яких відхилень від стандартів та планів. У загальному прийнятому циклі контролю моніторинг виконується шляхом оцінки результатів і відхилень, перевірки специфікації за кожним набутим результатом.

Процес управління якістю в проєкті включає:

1. Концепцію управління якістю в проєкті:

- вироблення стратегії управління якістю в проекті (визначення цілей і завдань, критеріїв успіху і невдач, обмежень і припущень);
- визначення загальних вимог і принципів забезпечення якості, стандартів і правил;
- вимоги до системи управління якістю;
- затвердження концепції.

2. Планування управління якістю в проекті:

- уточнення цілей, завдань, критеріїв оцінки та обмежень при управлінні якістю;
- визначення списків об'єктів контролю в проекті;
- опис продукту проекту, що впливає на планування якості;
- визначення показників оцінки якості на основі міжнародних, державних, галузевих і внутрішніх стандартів;
- розробка процедур управління якістю та їх опис;
- вибір методів і засобів контролю та оцінки якості;
- розробка плану управління якістю в проекті, що описує систему управління якістю в проекті і реалізацію процедур з управління якістю проекту.

3. Організацію та здійснення контролю якості в проекті:

- організація і здійснення управління якістю в проекті;
- технічна підтримка контролю якості;
- контроль якості в проекті;
- формування звітів для оцінки виконання якості.

4. Аналіз стану та забезпечення якості в проекті:

- порівняння фактичних результатів проекту зі специфікаціями і вимогами;
- аналіз стану і прогресу якості в проекті протягом його життєвого циклу;
- технічна оцінка якості продукту проекту;
- процес перевірки відповідності наявних результатів контролю якості існуючим вимогам;
- формування списку відхилень;
- визначення необхідних коригувальних дій щодо забезпечення якості в проекті;
- рішення про проміжне прийняття;
- уточнення списків контролю об'єктів;
- коригувальні дії щодо забезпечення якості в проекті;

- документування змін.

5. Завершення управління якістю в проєкті:

- зведена оцінка якості результатів проєкту;
- рішення про завершальне приймання;
- список зауважень і претензій за якістю;
- вирішення спірних питань і конфліктів;
- оформлення документації та архіву;
- аналіз досвіду і формування бази знань з управління якістю.

Питання до розділу 12

1. Які задачі вирішує процес управління змінами в проєкті?
2. Складові, причини та види змін у проєкті?
3. Особливості управління змінами в проєкті?
4. Система контролю за змінами та склад процесу управління змінами?
5. Поняття «якість» та ключові аспекти якості?
6. Менеджмент якості проєкту?
7. Планування управління якістю проєкту?
8. Контроль якості проєкту?
9. Процес управління якістю в проєкті?

РОЗДІЛ 13. КОРПОРАТИВНА СИСТЕМА УПРАВЛІННЯ ПРОЄКТАМИ

13.1. Корпоративна система управління проєктами

Корпоративна система управління проєктами (КСУП) – це сукупність, що включає:

- методологію управління проєктами;
- методів вирішення окремих задач;
- інструменти та програмні заходи;
- вимоги, регламенти та процедури виконання робіт у КСУП;
- елементи організаційної структури, які об'єднані в єдину цілеспрямовану систему, що використовуються для управління як окремими проєктами, так поєднаними за певними ознаками групами проєктів (портфелями проєктів та програмами) [1, 7].

КСУП визначає системний і процесний підходи до управління проєктами, які застосовуються по відношенню до всіх проєктів підприємства, а також знання, навички, інструменти та методи, які застосовуються в процесах та процедурах, що здійснюються у рамках управління проєктами.

КСУП включає в себе тільки ті основні елементи організаційної структури, що задіяні у процесі управління проєктами, ролі і функції персоналу, що приймає участь в правлінні проєктами, та формалізовані взаємовідносини між учасниками цього процесу.

Основні компоненти КСУП підприємства можуть бути поділені на три групи [6]:

- *Методи і методологія системи управління проєктами* і, як головний документ, корпоративний стандарт управління проєктами.
- *Персонал* – група людей, що має відповідну підготовку по управлінню проєктами та діє у відповідності до єдиних правил і вирішує завдання, які регламентовані цими правилами (стандартами, регламентами та наказами);
- *Інструментальні засоби* – програмні та технічні засоби автоматизованого середовища, що створюють єдиний інформаційний простір для учасників проєктів та забезпечує реалізацію методології управління проєктами підприємства.

Впровадження КСУП має плануватися по напрямкам, що відображають головні види забезпечення проєктної діяльності.

У плані впровадження та розвитку КСУП мають бути визначені такі напрямки:

- **Методичне забезпечення** (наприклад, положення про роботу у КСУП, регламенти взаємодії, концепції, положення, стандарти та ін.).
- **Програмне забезпечення** (включає впровадження стандартного ПЗ, наприклад, MS EPM та розробку спеціального).
- **Організаційне забезпечення** (серед найважливіших задач – розгортання проектного офісу, як організаційного підрозділу підприємства, розробка та затвердження регламентуючих документів).
- **Адміністративні заходи** (розробка наказів щодо започаткування та організації проектної діяльності, мотивації учасників КСУП всіх рівнів та інше).
- **Навчання** – організація та проведення навчання у підрозділах по відповідним рівням відповідальності за проектами та за ролями в управлінні проектами.
- **Розповсюдження досвіду** (проведення наради, семінари, конференції, Web Portal, база знань КСУП та ін.).

Одним з головних підходів у впровадження проектної діяльності є повна і всебічна стандартизація проектної діяльності, основними елементами якої є корпоративна методологія управління проектами.

13.2. Корпоративна методологія управління проектами

Корпоративна методологія управління проектами є набором процедур та внутрішніх нормативних документів, що визначають склад та зміст цих процедур, а також сукупністю інструментів та методів управління проектами, які забезпечують реалізацію всіх проектів підприємства за єдиними правилами і стандартами.

Корпоративна методологія управління проектами визначає як процедури управління (ухвалення та відхилення рішень за проектами) на різних фазах життєвого шляху, так і вимоги до проектів в різних функціональних областях: фінансах, кадрах, термінах, ресурсах, ризиках, якості, постачанні та інше.

Корпоративна методологія управління проектами визначається виробничими та структурними особливостями діяльності підприємства.

Програмне забезпечення, що використовується для підтримки процесів управління проектами, вибирається згідно з вимогами корпоративної методології управління проектами.

Корпоративна методологія управління проектами має створюватись у рамках розбудови КСУП, як елементу діяльності підприємства. Задачі по розробці і

впровадженню корпоративної методології мають вирішуватись у рамках організаційних складових КСУП на усіх етапах.

До організаційно-функціональних елементів КСУП відносяться [6]:

- **проектний офіс** (як підрозділ у структурі підприємства – наприклад, – відділ управління проектами);
- **група управління проектом** як тимчасове формування, що забезпечує виконання одного певного проекту, для управління яким вона і започаткована).

Проектний офіс (ПО) – це окремий підрозділ підприємства, що виконує задачі з організації управління проектами, підготовки і впровадження методологічного, програмного та навчального забезпечення з управління проектами у рамках всього підприємства.

Основні цілі створення:

- Розробка, впровадження, підтримка та розвиток методології і єдиних стандартів управління і звітності за проектами у підприємстві.
- Впровадження та підтримка загальної для підприємства системи навчання та перепідготовки працівників для ефективної участі у проектній діяльності.
- Розподіл обмежених ресурсів між проектами, рішення спірних питань.
- Впровадження єдиної системи інформування зацікавлених сторін про хід роботи над проектами.
- Організація створення бази знань.
- Створення архівної бази завершених проектів.
- Здійснення процедур ефективного моніторингу і контролю планування і виконання в цілях оптимізації робіт за проектами в цілому по підприємству.

Основні функції ПО у КСУП:

- Участь у аналізі та виборі проектів до здійснення.
- Перевірка відповідності паспорту проекту діючим КСТ.
- Моніторинг ходу виконання проектів.
- Підготовка планових та позапланових звітів Керівництву Підприємства.
- Розгляд, аналіз і підготовка затвердження запитів на зміни в проектах.
- Організація та забезпечення виконання документообігу між Проектним

Офісом і іншими учасниками проекту.

- Ведення реєстру проектів.
- Архівація завершених і закритих проектів.
- Створення бази знань (шаблонів, методів і т.ін.).
- Аудит проектів.

- Актуалізація і своєчасне оновлення проектної методології.
- Розробка і підтримка процедур документообігу.
- Розробка положення про порядок звітності.
- Розробка положення про документообіг проектної документації.
- Розробка положення про порядок архівації завершених і закритих проектів.
- Організація проведення оперативного консультування та усунення проблем у використанні програмних засобів.
- Організація проведення навчання та перепідготовки працівників для ефективної участі у проектній діяльності.
- Організація та проведення семінарів та нарад для поширення досвіду впровадження проектної діяльності.

Групи управління проектами (ГУП) – тимчасові утворення, що створюється для управління виконанням кожного конкретного проекту за поданням керівника ПО та при узгодження з керівником підрозділу, до якого відноситься цей проект.

Основні функції діяльності ГУП:

- Підготовка паспорту проекту та іншої необхідної документації.
- Контроль відповідності ходу виконання плану проекту.
- Оперативний моніторинг ходу виконання проекту.
- Підготовка регулярних оперативних звітів про хід виконання проекту.
- Підтримка єдиного стандарту і методології управління та звітності за проектами.
- Розподіл обмежених ресурсів між проектами, рішення спірних питань.
- Підготовка запитів на зміни в проекті та внесення санкціонованих змін.
- Виконання документообігу по проекту.
- Підготовка на архівацію завершених і закритих проектів.
- Підготовка звітів по завершених і закритих проектах.
- Запит та залучення відсутніх ресурсів.

13.3. Реєстр проектів

Реєстр проектів є основним, впорядкованим і цілісним інформаційним сховищем, що має надавати повну інформацію про всі види проектів, що велись, ведуться та можливо будуть вестись у рамках КСУП [1, 7].

Структура опису проектів у реєстру має забезпечувати повну і достовірну інформацію з проектів. При цьому інформаційна повнота опису проектів у реєстрі

залежить від структури реєстру і тому ця структура має бути достатньою за змістом та обсягом.

Таблиця 13.1 Агреговані групи проєктів

	Характеристики агрегованих груп проєктів	Типи агрегованих груп проєктів	
1	Типи груп агрегованих проєктів	Портфелі проєктів групи проєктів з ієрархічною структурою	Папки проєктів , групи проєктів, що пов'язані між собою певними ознаками
2	Ознаки , що використовуються для агрегації	Адміністративна структура підприємства – відношення проєктів до елементів структури	Не структуровані ознаки та їх комбінації
3	Результати агрегації	Статичні – повністю пов'язані зі структурною схемою підприємства	Динамічні – з'являються та існують тільки за потребою у аналізі стану виконання певної групи проєктів
4	Періодичність формування груп	Постійне – з уточнення по мірі виникнення нових проєктів	За потребою – по мірі виникнення потреби у тому чи іншому аналізі
5	Термін існування результатів агрегації	Постійне – з оновленням по мірі виникнення нових проєктів	Тимчасове – до закінчення проведення аналізу та потреби у його результатах
6	Форма результатів агрегації	Реальний портфель проєктів, що виконуються підприємством і супроводжуються у КСУП	Віртуальна папка проєктів, що є агрегованою групою побудованою за певними ознаками
7	Входження проєктів до групи	Одноразово – тільки до одного з портфелів підрозділів в ієрархії підприємства	Багаторазово – може входити до багатьох папок в залежності від ознак, за якими вони сформовані

Достовірність інформації при цьому цілком залежить від суб'єктивного фактору і тому має перевірятись шляхом планового аудиту. Реєстр проєктів є динамічною структурою, що відображає перелік проєктів у відповідності до їх

положення у портфелі проектів на поточний час. Реєстр проектів дозволяє також формувати папки проектів, склад та зміст яких формується безпосередньо користувачами КСУП у відповідності до їх потреб та ролей:

- Склад реєстру проектів може змінюватись та доповнюватись відповідно до поточних потреб та задач, що покладаються на КСУП.
- Для формування, аналізу та контролю реєстру процесів використовуються спеціальні програмні засоби, що забезпечують контроль за станом виконання окремих проектів, портфелів проектів та папок проектів.
- Портфелі та папки проектів є агрегованими групами проектів, що мають ряд характерних для них рис. Основні характеристики та особливості портфелів та папок проектів показані у таблиці 13.1.

13.4. Підтримка виконання проекту

Наведений нижче графічний опис моделі бізнес-процесу підтримки реалізації проектів (рис. 13.1) описує склад та послідовність виконання найбільш важливих робіт для основних груп виконавців цих робіт.

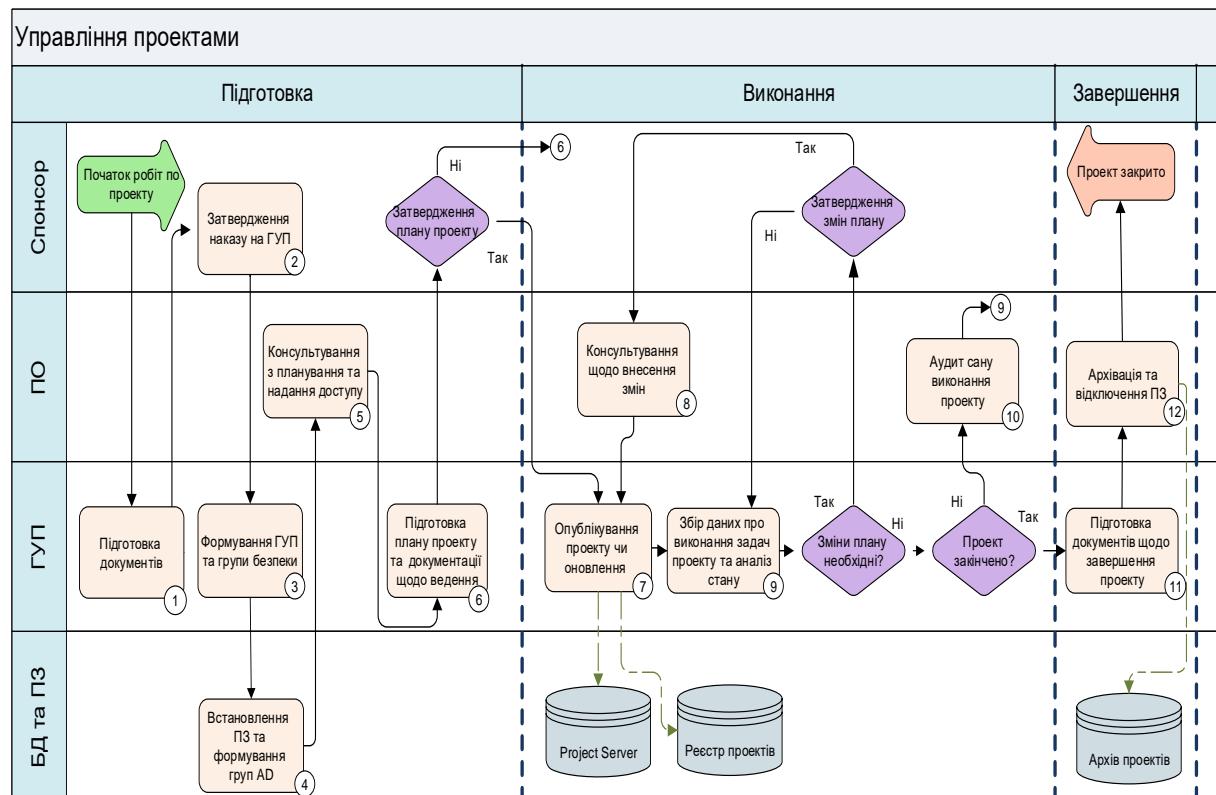


Рис. 13.1 Графічна модель підтримки виконання проектів (приклад)

Для скорочення розмірів опису в цій графічній моделі усі ролі, як для проектного офісу так і для ГУП, зведено у групові ролі. Роль проектного офісу (ПО) об'єднує ролі керівника, менеджера та адміністратора проектного офісу, а роль ГУП об'єднує ролі керівника та членів ГУП.

Модель відображає тільки найбільш важливі кроки і групи робіт, що мають виконуватись у процесі підтримки реалізації проектів. Більш детальна та повна модель процесу підтримки реалізації проектів у КСУП має бути описана у відповідному порядку (регламенті), що включає як опис послідовності робіт з посилання на їх виконавців, так і зразки та шаблони документів, які використовуються у цьому процесі.

У процесі підтримки виконання проектів, як показано у прикладі, виділяються такі основні кроки (рис. 13.1):

Крок 1. Керівник проєкту готує:

- обґрунтування проєкту (при необхідності для корпоративних проєктів);
- наказ на створення ГУП для конкретного проєкту (якщо є необхідність для складних проєктів).

Підготовлене обґрунтування проєкту та наказ на створення ГУП передаються на затвердження спонсору (відповідального за фінансування) проєкту та керівництву.

Функції членів ГУП можуть виконуватись у рамках службових повноважень адміністративних підрозділів без створення відповідних наказів.

У разі, коли ГУП не створюється, усі її функції виконує керівник проєкту.

Крок 2. Спонсор проєкту (керівництво товариства) затверджує наказ на ГУП, при необхідності створення такого формування для виконання проєкту, з працівників структурних підрозділів з частковим або повним відливом від основних штатних функцій на період виконання проєкту.

У наказі встановлюються основні задачі проєкту, строки виконання, керівник, координатор проєкту, перелік основних членів ГУП, що мають виконувати задачі проєкту з уточнення, при необхідності, конкретних функцій та ролей. Ці призначення на ролі можуть бути визначені і пізніше при уточненні паспорту проєкту та програми комунікацій.

Крок 3. Керівник проєкту чи уповноважений керівником координатор проводить необхідні наради з членами ГУП та дає роз'яснення і доручення щодо виконання проєкту.

Крок 4. Працівники СТП, що відповідають за супроводження програмних засобів MS Project, інсталюють ПЗ MS Project та створюють, при необхідності, групи в AD (active directory – активному каталогу).

Крок 5. Проектний офіс (ПО) надає права доступу до проекту на Project Server для членів ГУП та консультує членів ГУП щодо оформлення документів:

- деталізованого плану проекту;
- техніко-економічне обґрунтування проекту (за необхідністю);
- паспорту проекту (за необхідністю);
- план комунікацій та звітності щодо проекту (за необхідністю).

Крок 6. Керівник проекту чи уповноважений керівником координатор готовиться разом з членами ГУП план-графік проекту та інші необхідні документи щодо ведення проекту та передає їх на ухвалення (чи затвердження) спонсору проекту (керівництву товариства).

Крок 7. Керівник ГУП чи уповноважений керівником координатор опубліковує узгоджений (чи змінений та оновлений) план проекту на Project Server з використанням MS Project Professional.

Крок 8. Проектний офіс консультує щодо внесення змін у проект, при необхідності їх внесення.

Крок 9. Керівник та члени ГУП регулярно вносять дані про виконання робіт (циклічний процес) відповідно до затвердженого плану комунікацій та звітності щодо проекту.

Основними етапами цих робіт є:

- збір актуальних даних щодо стану виконання проекту від відповідальних за задачі членів ГУП;
- звітування згідно з планом комунікацій проекту, порядку та періодичності звітності членів ГУП;
- аналіз поточного стану, ризиків та підготовка оперативної інформації щодо необхідності внесення змін до проекту.

Якщо зміни потрібні, анкета змін готується та передається на затвердження спонсору, а в іншому разі проводиться перевірка на необхідність завершення проекту.

Якщо спонсор погоджується з необхідністю внесення змін, проектний офіс консультує щодо коректного внесення змін до проекту, що знаходиться у процесі виконання.

Крок 10. Якщо виконання проекту ще не закінчено, ПО проводить аудит виконання проекту, аналіз поточних результатів та актуальність існуючої інформації щодо виконання проекту за запитами спонсора проекту, керівництва чи регламентуючими документами.

Крок 11. ГУП готує документи на закриття (завершення) проєкту. Керівник проєкту відповідає за підготовку документів на закриття та архівацію. Остаточний звіт щодо виконання проєкту надається спонсору проєкту.

Крок 12. ПО за вимогою керівника проєкту архівує закінчений проєкт. Розміщені в архіві плани проєктів використовуються як прототипи та шаблони для наступних проєктів.

13.5. Особливості впровадження КСУП

За результатами досліджень PMI створення і розгортання КСУП на постійній основі в більшості організацій займає період від шести місяців до двох років. Значний вплив на строки і успішність впровадження КСУП у організації частіше за все впливають такі фактори [5, 8, 9]:

- Нечітке бачення моделі КСУП, його області застосування і результатів, які повинні бути отримані у керівництва компанії і відповідального за цей проєкт.
- Недостатня підтримка КСУП з боку керівництва організації.
- Укомплектованість КСУП співробітниками з низьким рівнем компетенції тільки за тих обставин, що вони наявні для призначення у КСУП.
- Недостатня кількість ресурсів, що виділяються для створення КСУП і формування необхідної інфраструктури для виконання офісом своїх функцій.

Одним із ключових показників при оцінці успішності та ефективності діяльності КСУП є цінність, яку він створює у рамках виконуваних процесів для таких зацікавлених сторін, як керівництво організації, сторони, що отримують результати або вигоди від проєктів, і менеджери проєктів. Вирішальним при оцінці цінності, створюваної КСУП для організації, як правило, виявляється думка менеджерів проєктів. Якщо КСУП не чинить їм необхідну підтримку у вирішенні проблем і в більш ефективному виконанні проєктів, то і у інших зацікавлених сторін, як правило, складається думка про низьку цінності від роботи КСУП. У зв'язку з цим керівник КСУП з моменту прийняття рішення про створення КСУП повинен виділити ключових для КСУП зацікавлених осіб і далі протягом усього життєвого циклу КСУП активно виявляти і управлюти їх очікуваннями від роботи КСУП, забезпечуючи надання цінних для них послуг на узгодженому рівні якості.

Питання до розділу 13

1. Що таке корпоративна система управління проєктами (КСУП)?
2. Які є основні компоненти КСУП?

3. Які є організаційно-функціональні елементи КСУП?
4. Які функції виконує проектний офіс?
5. Назвіть основні функції діяльності ГУП?
6. Що включає реєстр проектів?
7. Які є агреговані групи проектів?

РОЗДІЛ 14. МЕТОДОЛОГІЯ WATERFALL ТА AGILE. SCRUM ТА KANBAN

14.1. Waterfall та AGILE. Scrum та Kanban

Waterfall чи AGILE?

Найбільш поширені підходи в розробці software:

- методика Waterfall;
- методика AGILE;
- метод Scrum;
- метод Kanban.

Waterfall (водоспад) - методика управління проектами, яка має на увазі послідовний перехід з одного етапу на інший без пропусків і повернень на попередні стадії.

AGILE – методика управління різноманітними проектами, побудована **на системі ідей і принципів «гнучкого» управління проектами**, на основі яких розроблені популярні методи **Scrum, Kanban** і інші. Ключовий принцип - розробка через короткі ітерації (цикли), в кінці кожного з яких замовник (користувач) отримує робочий код або продукт.

Довідка.

Варіант 1. США

- Эджл, Amer. | ‘ædʒl | или | ‘ædʒəl |.
- З *наголосом на 1-й склад.*

Варіант 2. Британський.

- АджАйл, Brit. | ‘adʒaɪl | или | ‘ædʒaɪl |
- *Наголос на 1-й склад, але звучить, ніби на обоє склади.*
- *В україномовному просторі біль часто вживають 2-й варіант — аджАйл.*

14.2. Підхід до опису проектів Waterfall

Каскадна модель (англ. Waterfall model, іноді перекладають як модель «Водоспад») (рис. 14.1) - модель процесу розробки програмного забезпечення, в якій процес розробки виглядає як потік, що послідовно проходить фази:

1. Визначення вимог.
2. Проектування.
3. Конструювання (також «реалізація» або «кодування»).
4. Втілення.

5. Тестування та налагодження (також «верифікація»).
6. Інсталяція.
7. Підтримка.

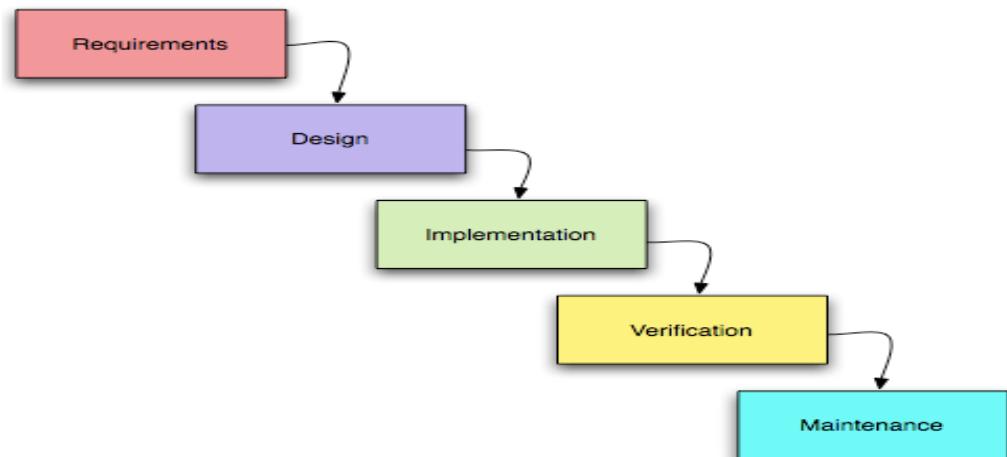


Рис. 14.1 Waterfall

Waterfall model засновано на статті У. Ройса опублікованій в 1970 році, в якій вказано можливість доопрацювання до ітеративної моделі розробки.

Каскадна модель має на увазі, що переход від однієї фази розробки до іншої відбувається тільки після повного і успішного завершення попередньої фази (рис. 14.2), і що переходів назад або вперед або перекриття фаз - не відбувається.

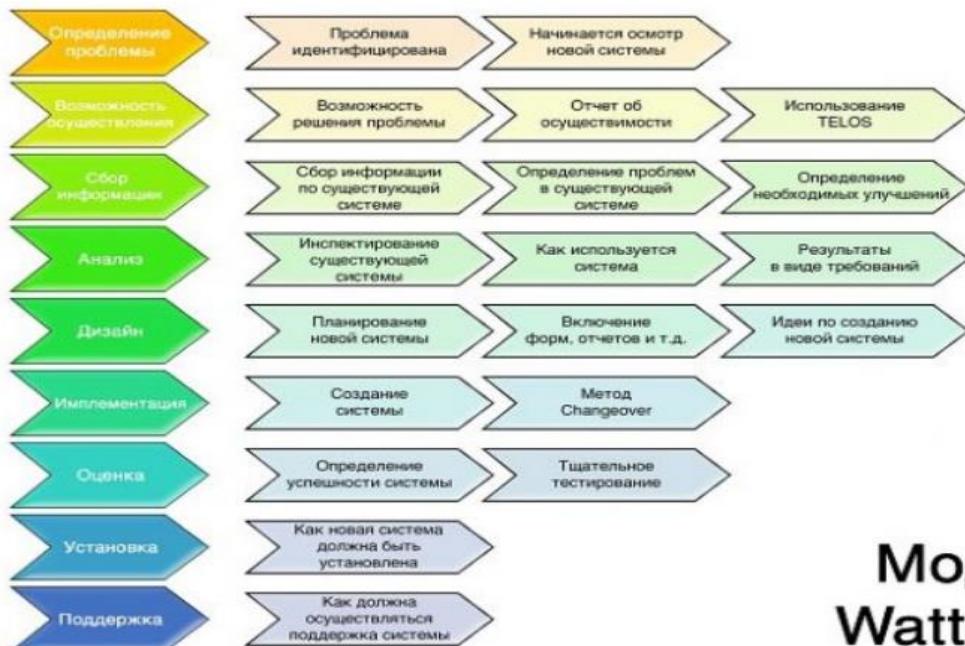


Рис. 14.2 Модель Waterfall

“-” (недоліки):

- недостатня гнучкість;
- як самоціль формальне управління проектом на шкоду термінам, вартості та якості.

“+” (переваги):

- В управлінні великими проектами формалізація є великою цінністю, так що може суттєво знизити ризики проекту та досягти більшої його прозорості.

В PMBOK 3-ї версії формально була закріплена лише методика «каскадної моделі».

Починаючи з PMBOK 4-ої версії крім «каскадної моделі» залучені і альтернативні – гнучкі ітеративні методи.

“+” Waterfall

- **Вартість і терміни** виконання зрозумілі ще до початку робіт. Тому замовник точно знатиме, коли проект завершиться і який бюджет потрібно витратити.
- **Інтуїтивно зрозуміла структура роботи**, як для досвідчених фахівців, так і для новачків.
- **Детально структурований план робіт** і продумана документація.
- Завдяки зручній звітності **легко відстежити витрачений час**, можливі ризики і використовувані ресурси в процесі роботи над проектом.
- **Завдання, які ставляться перед командою** ясні і не змінюються протягом усього проекту.
- **Якість проекту** займає **першочергове місце**, а витрачений час і бюджет відходять на другий план.

“-” Waterfall

- **Вимоги до проекту** закріплюються на початку і **не можуть змінюватися до закінчення робіт**. Цей факт позбавляє проект гнучкості.
- **Витрачається великий обсяг грошових коштів, часу і ресурсів**.
- **Неможливість** внесення змін у процесі розробки.
- **Замовник побачить готовий проект тільки після його релізу**, при необхідності змін можуть знадобитися додаткові кошти і час.
- **Взаємодія** між етапами розробки **повністю відсутня**.
- При використанні каскадної моделі **продукт тестується після його випуску**. Тому в більшості випадків проблеми виявляються тільки на етапі тестування.

Waterfall підійде, коли:

- Проектні вимоги ретельно продумані і незмінні. У замовника є **чітко сформульована концепція продукту**.
 - **Технології та інструменти відомі** заздалегідь.
 - Розроблюваний продукт **складний і вимагає великих витрат**.
 - **Пріоритетом є якість продукту.** Часові та грошові витрати мають другорядне значення.
 - **Замовник не планує брати участь в проекті.** Він бачить тільки готовий продукт. Проект повністю розробляється на аутсорсинг.
 - Клієнту **важливо знати точні терміни виконання всіх робіт** над проектом. Виконавець повністю несе відповідальність за зрив термінів і незаплановане збільшення бюджету.
 - **Реалізація аналогічного проекту, з яким вже стикалися розробники.**

14.3. AGILE – Гнучкі методи

Гнучка розробка ПЗ (англ. **Agile (гнучка) software development**), **agile-методи** - узагальнюючий термін для цілого ряду підходів і практик, заснованих на цінностях Маніфесту гнучкої розробки програмного забезпечення і 12 принципах, що лежать в його основі.

Маніфест гнучкої розробки програмного забезпечення (англ. **Agile Manifesto**) - основний документ, що містить опис цінностей і принципів гнучкої розробки програмного забезпечення, розроблений в лютому 2001 року на зустрічі 17 незалежних практиків декількох методів програмування, які іменують себе «**Agile Alliance**».

Текст маніфесту доступний на більш ніж 50 мовах (в т. ч. На українській), і включає в себе 4 цінності, 12 принципів.

4-и цінності в AGILE

1. **Люди і взаємодія** важливіше процесів та інструментів.
2. **Працюючий продукт** важливіше вичерпної документації.
3. **Співпраця з клієнтом** важливіше узгодження умов контракту.
4. **Готовність до змін** важливіше проходження попереднім планом.

Таким чином, не заперечуючи важливості того, що справа, ми все-таки більше цінуємо те, що зліва.

12 принципів AGILE

1. **Найвищим пріоритетом є задоволення потреб клієнта**, завдяки регулярному і ранньому постачанню коштовного програмного забезпечення.

2. Зміна вимог вітається, навіть на пізніх стадіях розробки.
3. **Працюючий продукт слід випускати якомога частіше**, з періодичністю від кількох тижнів до кількох місяців.
4. Протягом всього проєкту **розробники і замовники повинні щодня працювати разом**.
5. Над проєктом повинні працювати **мотивовані професіонали**. Щоб робота була зроблена, створіть умови, забезпечте підтримку і повністю довіриться їм.
6. **Безпосереднє спілкування** є найбільш практичним і ефективним способом обміну інформацією як з самою командою, так і всередині команди.
7. **Працюючий продукт** - основний показник прогресу.
8. Інвестори, розробники і користувачі повинні мати можливість **підтримувати постійний ритм** нескінченно.
9. Постійна увага до **технічної досконалості** і якості проєктування підвищує гнучкість проєкту.
10. **Простота** - мистецтво мінімізації зайвого клопоту - вкрай необхідна.
11. Найкращі вимоги, архітектурні та технічні рішення народжуються у **самостійно організованих командах**.
12. Команда повинна систематично аналізувати можливі способи поліпшення ефективності і відповідно **коригувати стиль своєї роботи**.

“+” AGILE

- **Внесення необхідних змін** і впровадження нового функціоналу може відбуватися **незалежно від циклу розробки продукту**, що значно підвищує конкурентні переваги готового проєкту.
- Проєкт **складається з коротких і зрозумілих циклів**, після закінчення яких клієнт отримує робочий продукт.
- **Гнучкий процес коригувань** в будь-який ітерації дозволяє знизити виробничі ризики.
- - Досить **швидкий реліз пробної версії** для подальших коригувань і тестування.
- **Високий ступінь зацікавленості всіх членів команди** і постійна взаємодія з замовником. Він завжди в курсі, на якій стадії знаходиться проєкт.
- **Показником ефективності є робочий продукт**, що вимагає високого професіоналізму від виконавців і грамотної організації робочого процесу.

“-” AGILE

- Розрахувати **кінцеві витрати практично неможливо** - вимоги можуть постійно змінюватися в залежності від особливостей проекту. Складність полягає в тому, що вони можуть суперечити вже існуючій структурі.

- **Agile вимагає великої зацікавленості в процес** і повного занурення в нього, що буває складно, особливо для молодих підрядників.

- **Можливість частого внесення правок** може обернутися ризиком в нескінченному вдосконаленні проекту. Тут також можлива і зворотна сторона - зниження якості продукту.

Коли застосовують AGILE (рис. 14.3):

- Коли перелік вимог достатньо не визначений, а зміни повинні вноситися максимально швидко.

- У проекті працює досвідчена команда з високим рівнем професіоналізму.

- Замовник бере активну участь в розробці протягом усього проекту.

- Клієнту важливо вносити зміни максимально оперативно на будь-якому етапі роботи.

- Якщо необхідно швидко і в короткі терміни створити робочу версію продукту.

- Новий проект є стартапом.

- Ніша, для якої розробляється продукт, схильна до постійних змін.

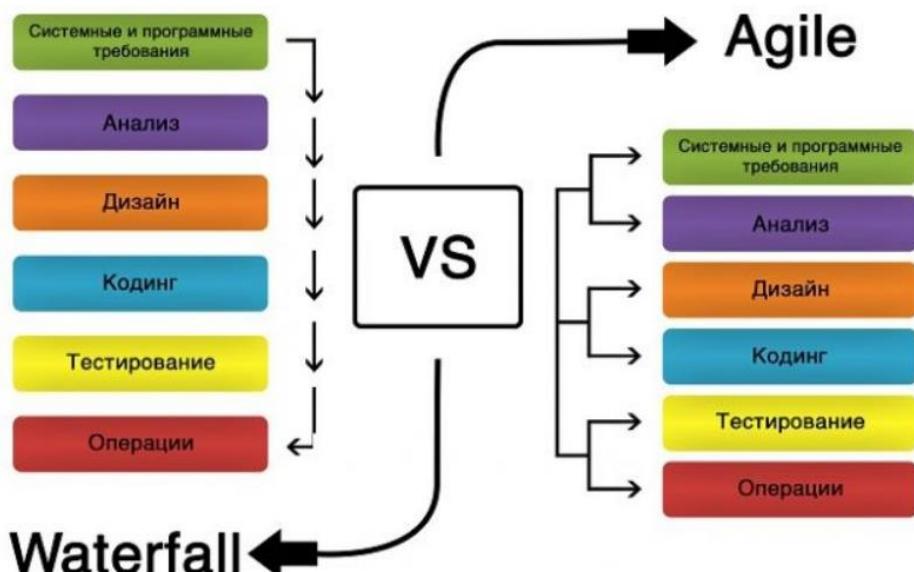


Рис. 14.3 AGILE проти Waterfall

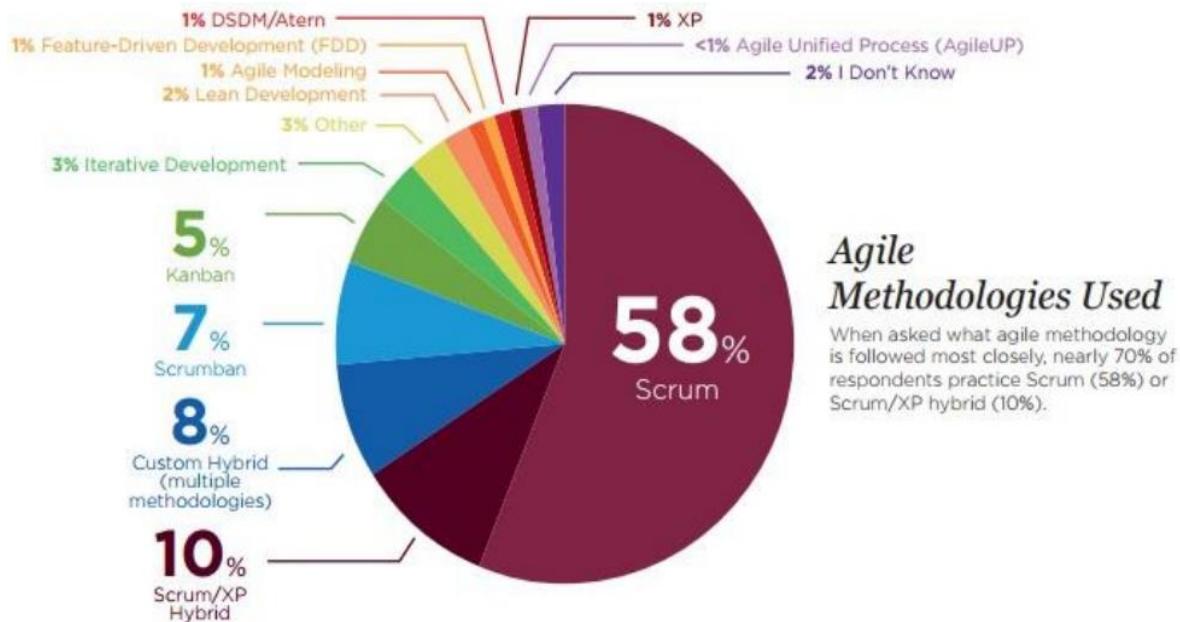


Рис. 14.4 Використання методології AGILE

Agile in the World

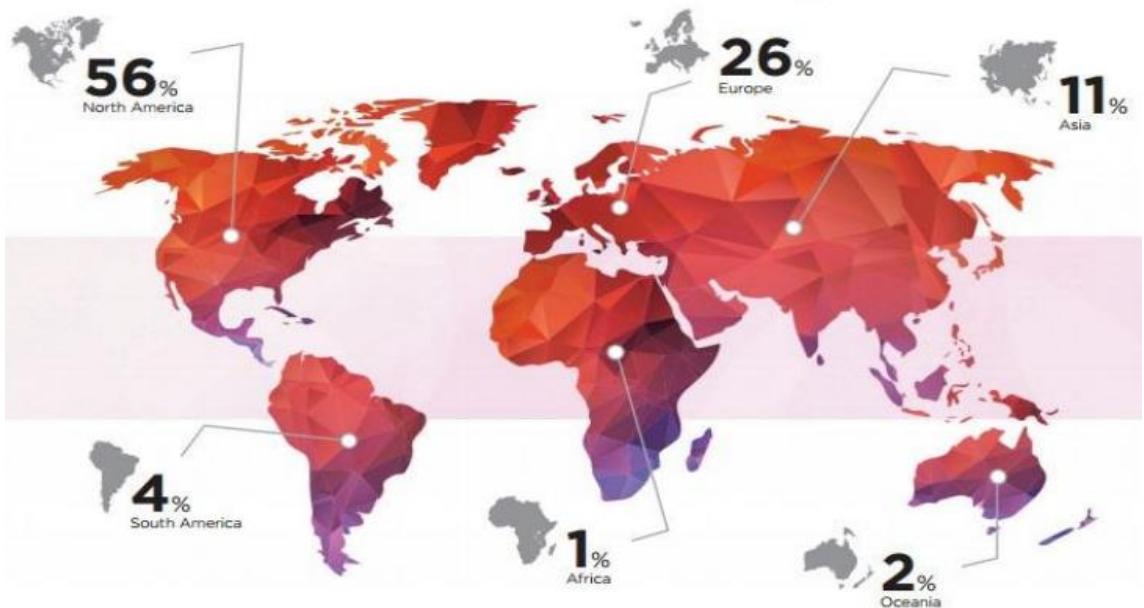


Рис. 14.5 Розповсюдження AGILE в світі

14.4. Порівняння Scrum та Kanban

Завдання команд **scrum** (з англ. – «сугучка») - поставити працююче ПЗ по частинам (складовим, компонентам) за ряд проміжків часу, які називаються спринтами (в англ. «sprint»).

Команди scrum прагнуть створювати цикли навчання для швидкого збору і обліку відгуків клієнтів. Scrum-команди використовують особливі ролі (таблиця 14.1), створюють спеціальні артефакти і проводять регулярні збори, щоб робота йшла в потрібному руслі. Найкраще методика scrum написана в керівництві по scrum.

Суть **kanban** (від яп. 看板 «рекламний щит, вивіска») - в візуалізації роботи, обмеження обсягу незавершеною роботи і досягнення максимальної ефективності (або швидкості).

Kanban-команди прагнуть максимально скоротити час, який витрачається на виконання проекту (або призначеної для користувача історії) від початку до кінця. Для цього вони використовують дошку kanban і безперервно вдосконалюють свій робочий процес.

Таблиця 14.1 Порівняння Scrum та Kanban

	Scrum	Kanban
Графік	Регулярні спринти з фіксованою протяжністю (наприклад, 2 тижні)	Неперервний процес
Підходи до релізу	В кінці кожного спринту	Не перервна поставка
Ролі	Власник продукту, scrum-майстер, команда розробників	Обов'язкових ролей нема
Ключові показники	Швидкість	Час виконання, час циклу, обсяг завершеної роботи (WIP)
Відношення до змін	Під час спринту команда не повинні вносити зміни	Зміни можуть відбутися у будь-який момент

14.5. Універсальна схема Scrum

Управління продуктом в Scrum складається з декількох універсальних для кожного контексту кроків (рис. 14.6).

1. Обирається «Власник продукту» - людина, яка стає сполучною ланкою між ринком (замовником продукту або кінцевим споживачем) і командою

виконавців. Ця людина відповідає за збільшення цінності продукту і зі старту бачить загальний задум.

2. **Збирається команда виконавців**, компетентність якої повинна поєднуватися з умінням злагоджено працювати.

3. Визначається **Scrum-майстер** (рис. 14.7). Тут майстер - це **адміністратор**, стежить за ходом роботи в команді, але не командувач, а допомагає в навченні, що забезпечує планове проведення зборів і т. ін.

4. **Створюється список вимог до мети і продукту** з розстановкою пунктів за пріоритетом. Цей список змінюється по ходу розвитку проєкту.

5. **Учасники команди оцінюють кожен пункт списку**, вирішуючи скільки **часових і матеріальних ресурсів** буде потрібно для реалізації завдання.

6. **Власник продукту, майстер і учасники команди проводять збори**, де в **спільному обговоренні планується спринт - короткий (щонайбільше місяць** в практиці розробників ПЗ) етап, на який планується рішення певної частини завдань. У деяких контекстах такий етап називають ітерацією. Передбачається, що протягом кожного спринту-ітерації команда напрацьовує певну кількість балів, яку в наступному спринті бажано збільшити, щоб продемонструвати зростання продуктивності.

7. Для інформування всіх учасників процесу **створюється інформаційна дошка** (рис. 14.8), що заповнюється стікерами, з поділом на те, що потрібно зробити, що знаходиться в роботі, і що зроблено. У міру виконання завдань, стікери переміщаються з однієї колонки в іншу.

8. **Короткі загальні збори проводяться щодня.** На них озвучуються перешкоди на шляху, визначається вже зроблене для користі проєкту і заплановане.

9. **Кожен спринт закінчується детальним оглядом результатів роботи і**, що не менш важливо, обговоренням характеристик процесу в минулому спринті. Якщо можна щось поліпшити, то обговорюються спрямовані на оптимізацію **нововведення, які будуть впроваджені в наступному спринті.**



Бэклог (або баклог) - це журнал роботи, що залишилась, яку необхідно виконати команді

Рис. 14.6 Схема процесу Scrum

Складові елементи Scrum. Scrum дошка.

Ролі	Артефакти	Процеси
<ul style="list-style-type: none"> • Власник продукту (Product Owner) • Скрам-майстер (Scrum Master) • Команда (Team) 	<ul style="list-style-type: none"> • Беклог продукту • Беклог спринту • Інкремент продукту 	<ul style="list-style-type: none"> • Планування спринту • Огляд Спринту • Щоденний Скрам

Рис. 14.7 Ролі, артефакти та процеси в Scrum

Пример Скрам-дошки						
Бэклог Продукта	Бэклог Спринта	В процессе	Взаимная проверка	На тести- ровании	Готово	Заблоки- ровано

Рис. 14.8 Приклад Scrum-дошки

Scrum був розроблений для проектів, в яких необхідні «швидкі перемоги» в поєднанні з толерантністю до змін. Крім того, цей метод підходить для ситуацій, коли не всі члени команди мають достатній досвід в тій сфері, в якій реалізується проект - постійні комунікації між членами командами дозволяють брак досвіду або кваліфікації одних співробітників за рахунок інформації і допомоги від колег.

Онлайн телеканал *Netflix* є відмінним прикладом швидких поставок результатів. Сайт ресурсу оновлюється кожні два тижні завдяки **Scrum**, який не просто дозволяє працювати з високою швидкістю, але й акумулює призначений для користувача досвід і дає можливість виявити найголовніше для клієнтів.

В ходіожної ітерації, розробники додають і тестують нові функції сайту і приирають ті, якими не користувалися клієнти. За словами команди *Netflix*, основна перевага **Scrum** в тому, що він дозволяє «швидко помилятися». Замість того, щоб довго і з великими витратами готовувати великий реліз, поставки раз в два тижні по **Scrum** мають невеликий розмір. Їх легко відстежувати і, якщо щось йде не так, швидко вправляти.

Scrum дуже вимогливий до команди проекту. Вона повинна бути невеликою (5-9 чоловік) і крос-функціональної - тобто члени команди повинні володіти **більш ніж однієї компетенцією, необхідної для реалізації проекту**. Наприклад розробник ПЗ повинен володіти знаннями в тестуванні і бізнес-аналітиці. Робиться це для того, щоб частина команди не «простоювала» на різних етапах проекту, а також для того, щоб співробітники могли допомагати і підміняти один одного.

Крім того, члени команди повинні **бути «командними гравцями**», активно брати на себе відповідальність і вміти самоорганізуватися. Підібрати таку зрілу команду дуже непросто!

Scrum підходить не для всіх команд і організацій ще й тому, що пропонований процес може не підійти для розробки конкретного продукту - **наприклад промислової програмно технічної системи.**

Scrum може допомогти коли:

- Концепція змінюється по ходу роботи.
- Потрібно запустити основу проекту з мінімальним бюджетом і термінами.
- Нервовий замовник - часто показуємо роботу.

Scrum не може допомогти якщо:

- Терміни/бюджет недостатні в принципі (не можуть допомогти і інші методики).
 - Команда не мотивована / недосвідчена.
 - Клієнт хоче все, на вчора і за безкоштовно.

Результати.

Для клієнта:

- Отримання найважливіших, з точки зору бізнесу, цінностей в найкоротші терміни.

Для команди:

- Ефективність.
- Творчість.
- Задоволення.

14.6. Можливості Kanban

Методологія kanban - це система постановки завдань, при якій всі етапи проекту візуалізують на спеціальній дошці. **Скільки етапів - стільки і стовпців в kanban-дошці** (рис. 14.9 – 14.11). Члени команди можуть бачити поточний стан завдання на будь-який момент часу. Це передбачає повну прозорість роботи.

Мета kanban - зробити проект **наочним, відстежити готовність робіт і проконтролювати навантаження фахівців.**

Для спрощення контролю робочий процес візуалізують на дошці, поділеної на колонки. Кожна колонка - це поточний стан робіт. Безпосередньо завдання відображають в **kanban-картках** - там можна прочитати їх опис, рівень важливості та додаткову інформацію. **Коли завдання завершує певний етап, картку з її описом переносять у відповідну колонку.** Поглянувши на дошку, можна відразу зрозуміти, яка ситуація з проектом.

Надо сделать	В работе	Выполнено
Задача 4	Задача 3	Задача 1
Задача 5		Задача 2
Задача 6		

Рис. 14.9 Kanban-дошка. Приклад 1

The screenshot shows a Trello board with four columns:

- Задачи**: Contains cards for 'Составление плана', 'Создание текстов по плану', 'Разработка дизайна к готовым текстам', 'Вёрстка писем', and a '+' button.
- В процессе**: Contains cards for '+ Добавить карточку' and a '+' button.
- На доработке**: Contains cards for '+ Добавить карточку' and a '+' button.
- На согласовании**: Contains cards for '+ Добавить карточку' and a '+' button.

Рис. 14.10 Kanban-дошка. Приклад 2

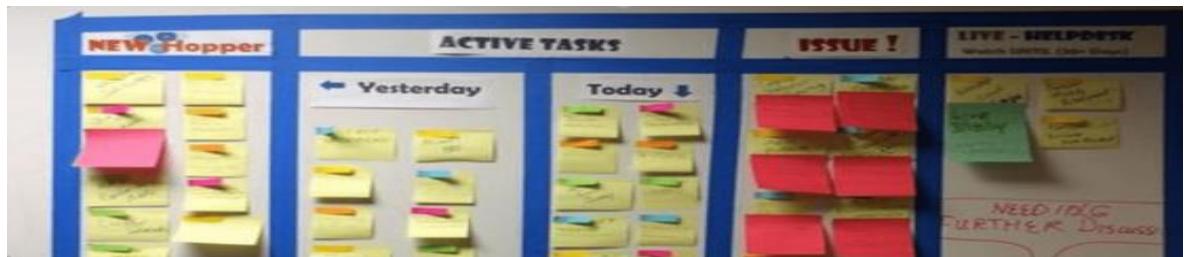


Рис. 14.11 Kanban-дошка. Приклад 3

Чотири базових принципи kanban

1. Відштовхуйтесь від того, що у вас є зараз.

Методологія не закликає **одномоментно** змінювати структуру компанії і ролі співробітників. Навпаки, потрібно **впроваджувати зміни у вже існуючу систему**.

2. Прагніти до поетапних, постійних і еволюційних змін.

Іншими словами - йти до великої мети маленькими кроками. На перший погляд може здатися, що глобальні зміни принесуть більше користі і прибутку. Але потрібно пам'ятати, що також вони принесуть величезні ризики.

Поступовий рух до мети - більш гнучкий і безпечний підхід.

3. Поважайте потокові процеси і ролі.

Потрібно зберегти те, що працює добре. Це стосується і відносин, і посад, і процесів. Зв'язки з людьми допоможуть отримати підтримку змін, а налагоджені процеси - удосконалити нестабільні.

4. Підтримувати лідерство на всіх рівнях

Прагнути бути лідерами і пропонувати зміни повинні співробітники на всіх рівнях, а не тільки менеджмент.

Принципи kanban

Візуальне відображення завдань. Всі завдання повинні бути представлені у вигляді карток і відображені на дошці. Дуже важливо оновлювати статус завдань. Наприклад, якщо розробники підготували код і передали в тестування, то картка із завданням повинна перейти до відповідного стовпця. Таким чином, будь-який учасник команди в будь-який момент часу може подивитися на якому етапі знаходиться завдання.

Обмеження по стовпцях WIP (work in progress або роботу, що виконується одночасно) на кожному етапі виробництва (**принцип один виконавець – одна задача!**). Щоб система рано чи пізно не "захлинулася" від потоку завдань, необхідно встановлювати обмеження. Наприклад, на kanban-дошці в стовпці Analysis (аналітика) у нас **працюють 2 людини і вони можуть обробляти не більше 2 задач**, немає сенсу навантажувати їх більше, так як наступні етапи системи будуть простоювати. Обмеження по стовпцях підбираються дослідним шляхом.

Фокус на невиконаних завданнях. Дивлячись на дошку із завданнями в першу чергу приділяйте увагу тим завданням, які "зависають" в тому чи іншому стовпці. Якщо у вас якийсь із етапів займає найбільше часу, то спробуйте перерозподілити ресурси або ж додати людей, якщо є така можливість.

Постійне поліпшення. Як тільки ви **врівноважите навантаження в системі**, вам буде простіше спостерігати за всім процесом в цілому. Виміряйте час циклу (скільки завдання висить в окремому стовпці, а скільки від моменту потрапляння в To do, до релізу Done). Міняйте навантаження в системі і скорочуйте час на проходження всіх стадій.

Приділяйте увагу дрібницям. Наприклад, якщо код, який пишуть розробники періодично не проходить тестування і повертається на доопрацювання, то можливо, є варіанти поліпшити якість розробки, щоб в тест потрапляв більш якісний продукт?

Переваги Kanban:

- Помилки видно одразу.
- Гнучкість планування.
- Швидка ідентифікація проблемних місць.
- Прозорість (всі бачать хто чим зайнятий).

Недоліки Kanban:

- Якщо в команді більше п'яти осіб, ефективність методології знижується.
- Довгострокове планування неможливе.

14.7. Порівняння Kanban та Scrum. Висновки

Хоч і **Kanban**, і **Scrum** відносяться до **AGILE**, між ними є кілька серйозних відмінностей.

По-перше, довжина ітерацій. Якщо в **Scrum** заздалегідь планується спринт (зазвичай тижневий або двотижневий), який не змінюється, то в **Kanban** нові завдання можна додавати коли завгодно. Основна мета **Kanban** - закрити завдання, а **Scrum** - закрити спринт.

По-друге, в **Scrum** завдання **оцінюють в Story points або в годинах**, щоб прорахувати, де буде команда після спринту. У **Kanban** розраховують **тільки середній час на виконання одного завдання**.

По-третє, **Scrum** більше підходить для старту проекту, тому що дозволяє точніше визначити терміни і тісно взаємодіяти з командою. **Kanban** ж більш популярний для підтримки вже готового продукту, його розвитку та модернізації - в ньому менше комунікації з командою, а завдання приходять в непередбачуваною послідовності.

Питання до розділу 14

1. Найбільш поширені підходи в розробці software?
2. Що становить методика «водоспад». Її фази?
3. Плюси та недоліки методики «водоспад»?
4. Методика гнучких методів. Її принципи та цінності?
5. Плюси та недоліки методики гнучких методів?

6. Порівняльні відмінності методики «водоспад» та методики гнучких методів?
7. Схема та складові методики Scrum?
8. Сильні та слабкі сторони методики Scrum?
9. Методика Kanban. Схема та складові?
10. 4 базові принципи Kanban?
11. Сильні та слабкі сторони Kanban?
12. Надайте порівняльну характеристику методам Scrum та Kanban?

РОЗДІЛ 15. СТАНДАРТ ISO 21500

15.1. Особливості стандарту ISO 21500

В 2012 році був опублікований стандарт з управління проєктами ISO 21500, затверджений на даний момент США і Євросоюзом. Цей стандарт значно коротший від попередніх, але розглядає проєкт не тільки як окрему діяльність, а і як складову частину загальної діяльності організації а також більш об'ємних сутностей, таких як портфелі і програми.

Цілями стандарту ISO 21500 є надання інформації вищому керівництву компаній про принципи та практику управління проєктами, забезпечення керівників проєктів і членів команди проєкту актуальними стандартами і практиками, забезпечення розробників національних і корпоративних стандартів базовим документом.

Цей міжнародний стандарт є загальним довідником для понять і процесів управління проєктами, котрі мають суттєвий вплив на досягнення результатів проєктів.

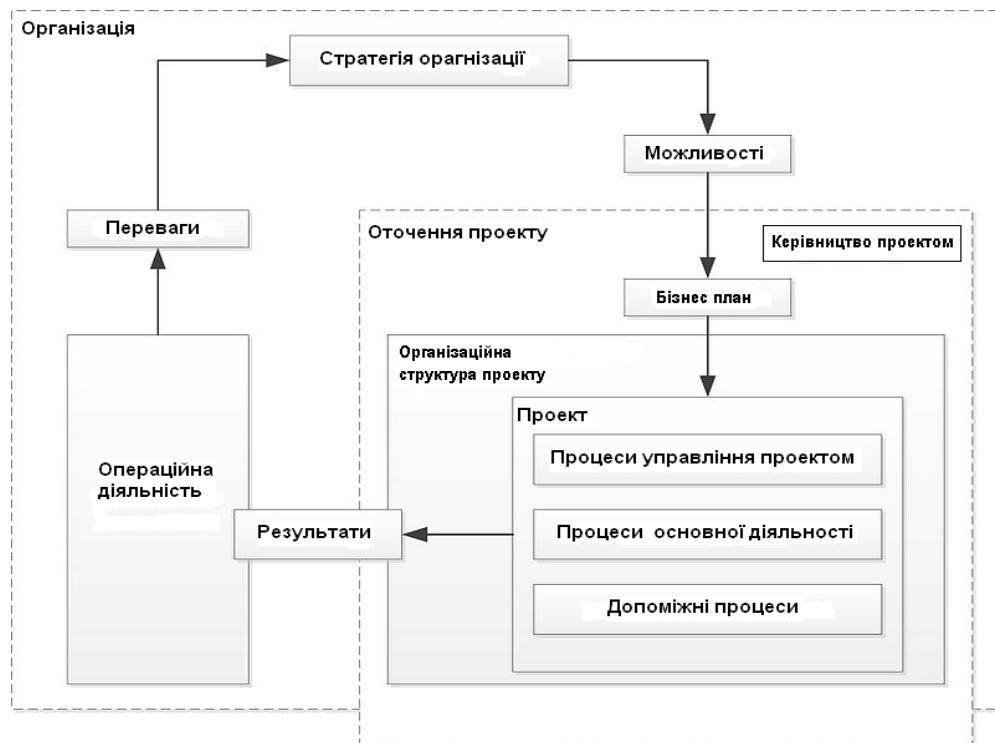


Рис. 15.1 Зв'язок між концепціями управління проєктами

Цільовою аудиторією є топ-менеджери і спонсори проєкту, керівники проєктів і члени команди проєкту, а також розробники національних і корпоративних стандартів.

В цьому стандарті проєктна діяльність тісно пов'язана з стратегією і операційною діяльністю всієї організації.

На рис. 15.1 показано, як концепції управління проєктами зв'язані між собою. Організаційна стратегія визначає можливості. Можливості оцінюються і мають бути задокументовані. Обрані можливості надалі розробляються у бізнес-план або інший аналогічний документ, на основі якого ініціюється один або кілька проєктів, що приносять результати. Ці результати можуть використовуватись для отримання вигоди. Вигода може бути використана в якості входного параметру для реалізації і подальшого розвитку організаційної стратегії.

Організації затверджують стратегію, що основана на місії, баченні та політиці. Звичайно, проєкти підпорядковуються стратегічним цілям. На рис. 15.2 приведено типовий цикл управління портфелем проєктів від стратегії до отримання вигід.



Рис. 15.2 Управління портфелем проєктів від стратегії до отримання вигід.

15.2. Процеси проєкту в ISO 21500

В ISO 21500 проєкт визначається як унікальний набір процесів, що включає координовані контролювані операції з датою початку і завершення, що виконуються для досягнення цілі.

Будь-який проєкт має визначені початок і завершення. Звичайно проєкт реалізується через послідовність фаз.

Далі в стандарті визначається управління проєктом як застосування методів, інструментів, технік і компетенцій до проєкту. Управління проєктами включає інтеграцію різних фаз життєвого циклу проєкту.

Процеси, обрані для використання в проєкті, будуються на системній основі. Результати проєкту регулярно оцінюються в ході його реалізації для відповідності вимогам куратора, замовника і інших зацікавлених учасників.

На базі поточних можливостей організації, може бути розроблений перелік можливостей. Ці можливості можуть бути оцінені для підтримки свідомого прийняття

рішень керівництвом для виявлення проєктів які змогли б перетворити деякі або всі ці можливості в вигоди.

Такі можливості можуть відповідати, наприклад, новим потребам ринку, актуальною організаційною потребою або вимогам регуляторів. Зазвичай організації знаходять і призначають Куратора проєкту для балансування цілей проєкту і вигоди.

Цілі проєкту та вигоди досягаються в результаті визначення проєктних інвестицій, наприклад, у вигляді бізнес-плану і це може бути підставою для вибору пріоритетів при виборі можливостей. Завданням визначення інвестицій є затвердження менеджментом інвестицій в проєкт.

Процес оцінки може включати безліч критеріїв, включаючи інвестиційну і якісну оцінку, наприклад, відповідність стратегічним цілям, соціальне значення або вплив на навколошнє середовище і може відрізнятися для різних проєктів.

Реалізація вигід є відповідальністю менеджменту організації замовника, яка може використовувати результати проєкту для отримання вигід у відповідності зі стратегією організації. Менеджеру проєкту слід враховувати вигоди і їх реалізацію бо вони будуть впливати на прийняття рішень протягом життєвого циклу проєкту.

Команда проєкту повинна враховувати наступне:

Зовнішні чинники поза межами материнської організації - соціально-економічна, географічна, політична, правова, технологічна і екологічна ситуація.

Внутрішні чинники всередині материнської організації - стратегія, технології, проєктна організаційна зрілість і доступність ресурсів, корпоративна культура і організаційна структура.

15.3. Учасники проєктів, програм та портфелів проєктів

Проект зазвичай виконується всередині організації, що охоплює й інші види діяльності. У таких випадках, існують відносини між проєктом та його оточенням, бізнес-плануванням та операційною діяльністю. Перед проєктна та після проєктна діяльність може включати в себе такі заходи, як розвиток бізнес-плану, проведення техніко-економічного обґрунтування і передача результатів проєкту в операційну діяльність. Проєкти можуть бути організовані в рамках програм та портфелів.

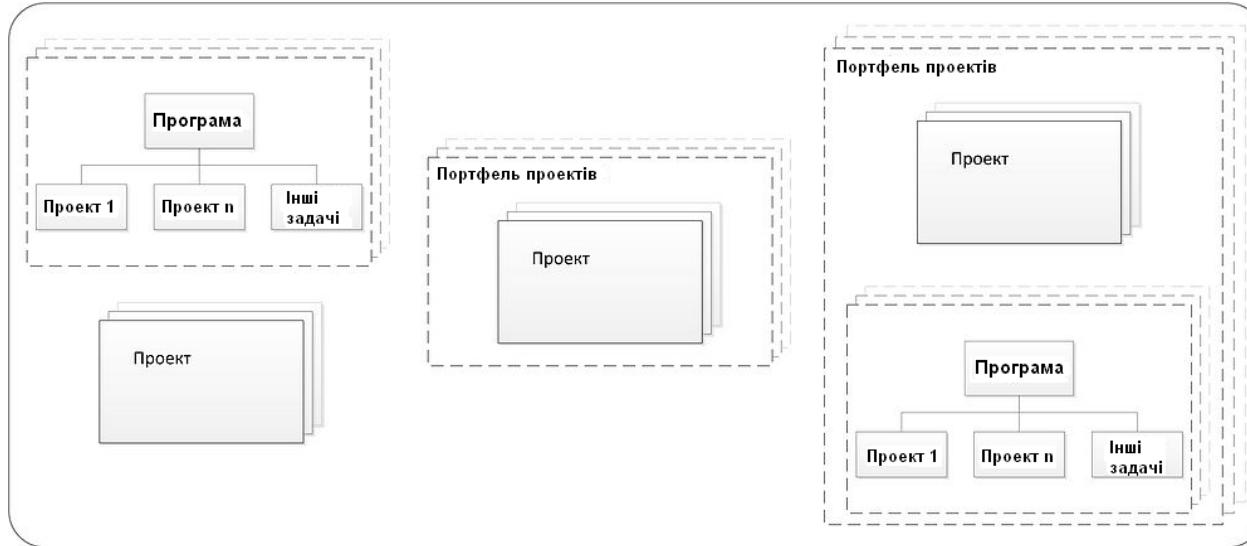


Рис. 15.3 Проєкти, програми і портфелі проєктів

Портфель проектів - це набір проектів та програм і окремих робіт, які згруповані разом в цілях сприяння ефективному управлінню для досягнення стратегічних цілей організації. Управління портфелем проектів - це централізоване управління одним або декількома портфелями проектів, яке включає виявлення, встановлення пріоритетів, затвердження, управління і контроль проектів, програм та інших робіт для досягнення конкретних стратегічних цілей.

Може виявитися доцільним проведення виявлення і вибору можливостей, затвердження і управління проектами за допомогою автоматизованої системи управління портфелями проектів.

Програма - це група пов'язаних проектів і окремих робіт, що відповідають стратегічним або іншим важливим цілям. Управління програмами полягає в централізованій і скоординованій діяльності по досягненню цих цілей.

Крім управління проектом (project management) в стандарті вирізняється зовнішнє керівництво проектом (project governance), яке включає такі аспекти, як визначення структури управління; політика, процеси та методології, які будуть застосовуватися; обмеження в повноваженнях при прийнятті рішень; відповідальність та підзвітність зацікавлених учасників; взаємодії такі, як звітність та ескалація проблем чи ризиків. Організація функціонує з метою досягнення конкретних цілей. В цілому вся діяльність організації може бути розділена на проектну і операційну. Операційна діяльність відрізняється від проектної в першу чергу тим, що вона здійснюється відносно постійними командами протягом повторюваних процесів і

націлена на підтримку життєздатності організації. Проєкти ж реалізуються тимчасовими командами, не повторюються і створюють унікальні результати.

Крім самої команди і керівництва проєкту існує ще ряд учасників проєкту, яких прийнято називати зацікавленими сторонами (stakeholders). Для підвищення ймовірності успіху проєкту, зацікавлені сторони проєкту, в тому числі організація, в якій виконується проєкт, повинні бути достатньо детально описані. Ролі та відповідальність зацікавлених сторін можуть визначатися у зв'язку з цілями проєкту та організації. Стандартні зацікавлені сторони проєкту показані на рис. 15.4.

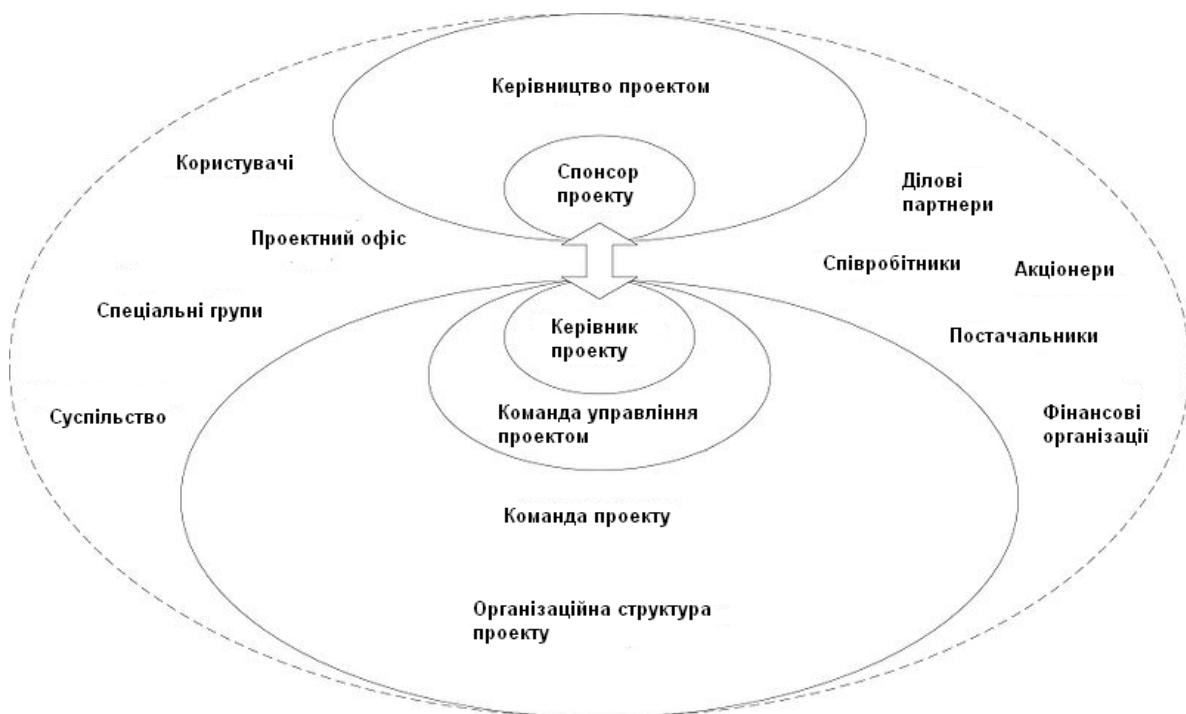


Рис. 15.4 Зацікавлені сторони проєкту (Project Stakeholders)

Оргструктура проєкту - це тимчасова структура, яка включає ролі, обов'язки, межі та рівень повноважень в проєкті, які повинні бути визначені і доведені до відома всіх зацікавлених сторін проєкту. Оргструктура проєкту може залежати від правових, комерційних, міжвідомчих або інших угод, які існують між зацікавленими сторінами проєкту, і може включати наступні елементи:

- менеджер проєкту - керує і управляє роботами проєкту і несе відповідальність за досягнення результатів проєкту;

- команда управління проектом (при необхідності) - надає допомогу менеджеру проекту в керівництві та управлінні роботами проекту та досягненні результатів проекту;
- команда проекту - виконує роботи проекту для успішного завершення проекту.

Зовнішнє управління проектом з боку Замовника (Project governance) може включати в себе наступних осіб:

- Спонсор проекту - очолює, схвалює старт проекту, виділяє ресурси, полегшує і забезпечує виконання проекту. Приймає виконавчі рішення та вирішує проблеми і конфлікти, що виходять за межі повноважень менеджера проекту.
- Керівний комітет або рада (при необхідності) - робить свій внесок у проект, що забезпечує вищий рівень керівництва проекту.

Рис. 15.4 включає в себе наступних додаткових зацікавлених осіб:

- Замовник або представник Замовника - вносять вклад у проект, за допомогою формування вимог до проекту та прийняття результатів проекту;
- Постачальники - вносять вклад у проект шляхом надання ресурсів для реалізації проекту;
- Проектний офіс - може виконувати широкий спектр заходів, включаючи управління, стандартизацію, навчання управлінню проектами, планування і моніторинг проекту.

Суттєвим питанням, що зумовлює успіх проекту, є компетентність учасників проекту. Підвищення рівня компетентності досягається навчанням, тренуванням і наставництвом всередині або за межами організації.

Проекти, як правило, організовані в фази, які визначаються потребами управління і контролю. Фази проекту складають життєвий цикл проекту. Життєвий цикл проекту охоплює період від початку проекту до його кінця.

Для кожного проекту існує перелік обмежень, в межах яких він виконується.

Існує кілька типів обмежень і, так як обмеження часто взаємозалежні, для менеджера проекту важливо врівноважити окремі обмеження з іншими. Результати проекту повинні відповідати вимогам проекту, і задовольняти обмеженням, таким як зміст, якість, терміни, ресурси і вартість. Обмеження, як правило, взаємопов'язані так, що зміна одного може вплинути на одне або декілька інших обмежень. Таким чином, обмеження можуть вплинути на рішення, що приймаються в рамках процесів управління проектами.

Досягнення консенсусу між ключовими учасниками проєкту з обмежень можуть сформувати міцний фундамент для успіху проєкту.

Цей стандарт визначає рекомендовані процеси управління проєктом, які можуть використовуватися в ході виконання проєкту в цілому, окремої його фази або обох. Ці процеси управління проєктом підходять для проєктів будь-яких організацій. Управління проєктом вимагає істотної координації і, як таке, вимагає, щоб кожен процес був би вивірений і пов'язаний з іншими процесами для забезпечення успіху всього проєкту. Деякі процеси можуть виконуватись кілька разів для більш повного визначення відповідності вимогам зацікавлених сторін і узгодження цілей проєкту.

Керівникам проєкту спільно з іншими зацікавленими сторонами проєкту рекомендується уважно вивчити процеси, наведені в стандарті ISO 21500, щоб застосовувати саме ті процеси, які відповідають проєкту і потребам організації.

15.4. Процеси управління проєктами

Процеси управління проєктами можна розглядати в двох різних ракурсах: з погляду управління проєктом (process groups) і з точки зору групування процесів за предметними галузями (subject groups). Ці дві різні групи представлені в таблиці 15.1. Самі процеси, незалежно від групування залишаються незмінними.

Таблиця 15.1 Відповідність процесів управління проєктами групам процесів і предметним групам

Предметні групи (subject groups)	Групи процесів (process groups)				
	Ініціація	Планування	Виконання	Управління	Завершення
1	2	3	4	5	6
Інтеграція	Розробка уставу проєкту	Розробка планів проєктів	Безпосередня робота за проєктом	Управління проєктними роботами	Закриття окремої фази або проєкту
				Управління змінами	Отримані знання
Зацікавлені сторони	Визначення зацікавлених сторін		Управління зацікавленими сторонами		

Продовження табл. 15.1

1	2	3	4	5	6
Зміст		Визначення змісту проєкту		Управління змістом проєкту	
		Створення структури декомпозиції робіт			
		Визначення складу робіт			
Ресурси	Створення команди проєкту	Оцінка ресурсів	Розвиток команди проєкту	Управління ресурсами	
		Визначення організаційної структури проєкту		Управління командою проєкту	
Час		Послідовність робіт		Управління розкладом	
		Оцінка тривалістю робіт			
		Розробити розклад			
Вартість		Оцінка витрат		Управління витратами	
		Розробка бюджету			
Ризики		Визначення ризиків	Ставлення до ризиків	Управління ризиками	
		Оцінка ризиків			

Продовження табл. 15.1

1	2	3	4	5	6
Якість		План по якості	Забезпечення вимог якості	Управління якістю	
Поставки		План поставок	Вибір постачальників	Адміністрування контрактів	
Комунікації		План комунікацій	Розповсюдження інформації	Управління комунікаціями	

Процеси ініціації (Initiating process group) використовуються для запуску фази проєкту або проєкту, для визначення мети фази або проєкту в цілому, з метою уповноважити керівника проєкту приступити до роботи над проєктом.

Процеси планування (Planning process group) використовуються для детального планування. Ця деталізація повинна бути достатньою для встановлення базового плану, на підставі якого контролюється виконання проєкту.

Процеси виконання (Implementing process group) застосовуються для виконання робіт з управління проєктами і забезпечують досягнення результатів, визначених у плані проєкту.

Процеси контролю (Controlling process group) потрібні для відстеження, аналізу і регулювання ходу і ефективності виконання проєкту відповідно до плану. Отже, у разі необхідності можуть бути зроблені попереджувальні та коригувальні дії, і виконані запити на зміни в цілях досягнення цілей проєкту.

Процеси завершення (Closing process group) виконуються для формального завершення проєкту або фази і проведення аналізу накопичених знань з метою їх застосування в майбутньому.

Управління проєктом починається з групи процесів ініціації і закінчується групою процесів завершення. Взаємозалежність між групами процесів вимагає взаємодії групи процесів управління з усіма іншими групами процесів. На рис. 15.5 показані взаємодії між групами процесів всередині проєкту, включаючи основні входи і виходи груп процесів. За винятком групи процесів контролю, групи процесів пов'язані послідовно. У той же час, групу процесів контролю можна вважати самостійною, оскільки її процеси використовуються для контролю не тільки проєкту в цілому, але й окремих груп процесів, як це показано на рис. 15.5.

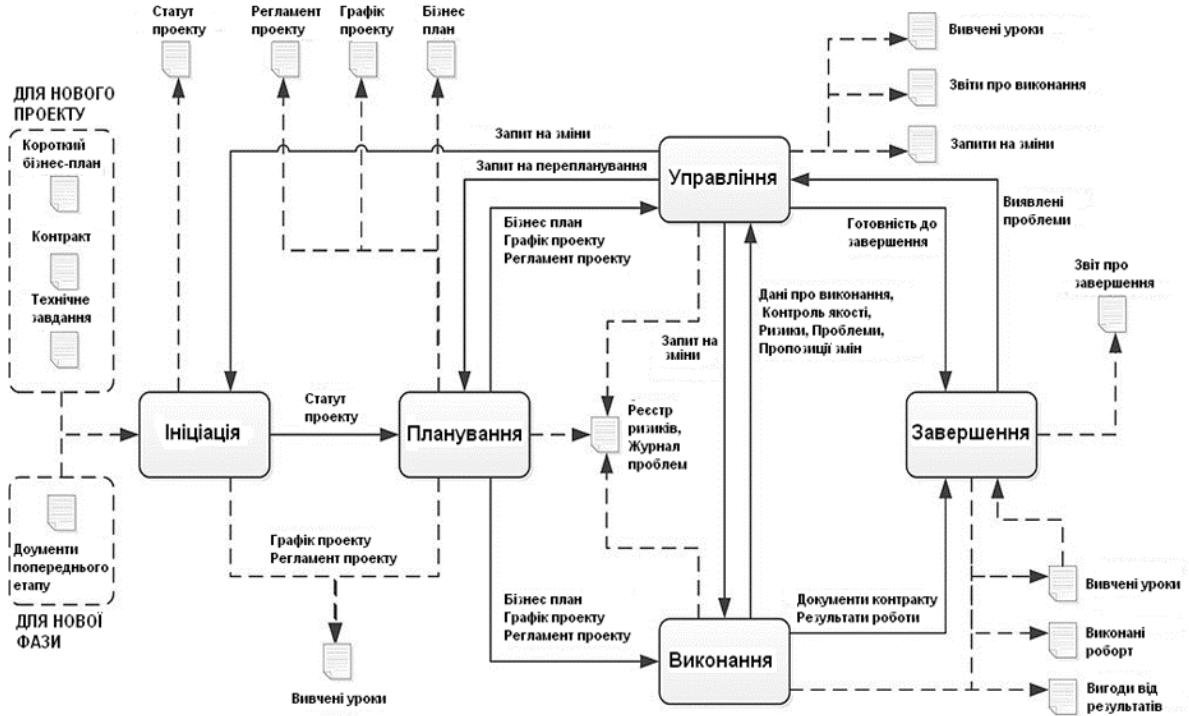


Рис. 15.5 Взаємодія груп процесів із вказаними основними входами і виходами

Предметна група інтеграції (Integration) включає в себе процеси, необхідні для ідентифікації, визначення, комбінації, уніфікації, координації, контролю та завершення різних видів діяльності і процесів, пов'язаних з проектом.

Предметна група зацікавлених сторін (Stakeholder) включає в себе процеси, необхідні для виявлення та управління спонсором проекту, споживачами та іншими зацікавленими сторонами.

Предметна група змісту (Scope) включає в себе процеси, необхідні для ідентифікації і визначення робіт і результатів, а також тільки необхідну роботу і результати.

Предметна група ресурсів (Resource) включає в себе процеси, необхідні для виявлення і придбання необхідних ресурсів проекту, таких як люди, приміщення, обладнання, матеріали, інфраструктура та інструменти.

Предметна група часу (Time) включає в себе процеси, необхідні для планування діяльності за проектом та контролю за ходом виконання проекту, з метою керування розкладом.

Предметна група вартості (Cost) тема включає в себе процеси, необхідні для розробки бюджету та спостереження за прогресом для управління витратами.

Предметна група ризиків (Risks) включає в себе процеси, необхідні для ідентифікації і управління загрозами і можливостями.

Предметна група якості (Quality) включає в себе процеси, необхідні для планування і забезпечення та контролю якості.

Предметна група закупівель (Procurement) включає в себе процеси, необхідні для планування та придбання продуктів, послуг або результатів, а також для управління взаємодій з постачальниками.

Предметна група комунікацій (Communication) включає в себе процеси, необхідні для планування, управління та розповсюдження інформації, що має відношення до проєкту.

В стандарті ISO 21500 також наводиться детальний опис кожного окремого процесу, що входить до процесів управління проєктами (див. Табл. 15.1), і його роль в управлінні проєктами.

Необхідно зауважити, що процеси управління проєктами не є задачами проєктів. Вони являють собою метаструктуру управління будь-якими проєктами і не є специфічними для окремого проєкту.

Питання до розділу 15

1. Призначення та можливості стандарту ISO 21500?
2. Концепція стандарту ISO 21500?
3. Визначення цілей та вигід у концепції стандарту ISO 21500?
4. Опишіть дві концепції представлення процесів управління у стандарті ISO 21500?
5. Опишіть управління портфелем проєктів у стандарті ISO 21500
6. Учасники проєкту визначені стандартом ISO 21500?
7. Опишіть представлення зовнішнього керівництва проєктом (project governance) у стандарті ISO 21500?
8. Опишіть предметні групи та групи процесів та їх співвідношення визначені стандартом ISO 21500?

РОЗДІЛ 16. СИСТЕМИ КЕРУВАННЯ ВЕРСІЯМА GITHUB.

16.1. Загальні задачі систем керування версіями

Множина вимог до ІТ проектів ознак якісного ПЗ: надійність, стійкість, модульність, безпечність, продуктивність, масштабованість, зручність застосування, тестування та супровід, і багато інших.

При цьому проект має включати: гарну документацію, включаючи журнали змін; деталізовані узгодження та стандарти; версіонування; автоматизовані тести та ін. Одним з зручних та ефективних засобів покращення робочих процесів стали системи верифікування версій, що на сьогодні надають гнучкі інструменти, які не диктують вам як потрібно реалізувати та документувати зміни, яку стратегію злиття використовувати, на яких етапах та стадіях фіксувати результати, які інструменти застосовувати, яким стандартам commit-ів (фіксування змін) слідувати та що потрібно рецензувати. Все це зорієнтовано вашими творчими підходами.

16.2. Що таке система контролю версій

Контроль версії - це практика відстеження змін програмного коду та управління ним. Системи контролю версій — це програмні інструменти, які допомагають командам розробників керувати змінами в вихідному коді з течією часу. В умовах праці вони допомагають команді розробників працювати швидше і ефективніше. Системи контролю версій найбільш корисні для команд **DevOps**, оскільки допомагають скоротити час розробки та збільшити кількість успішних розгортань.

Програмне забезпечення контролю версій відстежує всі зміни, що вносяться в код, у спеціальній базі даних. При виявленні помилки розробники можуть повернутися назад і порівняти з попередніми версіями коду для виправлення помилок, зводячи до мінімуму проблеми для всіх учасників команди.

Система контролю версій (СКВ від англ. Version Control System, VCS) - це спеціалізоване місце зберігання коду, система, що записує зміни у файл, чи набір файлів протягом часу і дозволяє повернутися пізніше до певної версії [15].

Інше формулювання звучить так:

СКВ - це програмне забезпечення, яке допомагає розробникам програм співпрацювати та вести повну історію своєї роботи. Воно може зберігати зміни файлів та модифікації вихідного коду. Щоразу, коли користувач вносить зміни в проект, СКВ приймає стан проекту та зберігає їх. Ці різні збережені стани проекту відомі як версії.

Системи **VCS** інколи називають інструментом **SCM** (source code management - управління початковим кодом) або **RCS** (revision control system - система управління редакціями).

Усі системи контролю версій по суті вирішують 4 задачі.

1. **Доступ до коду.** Вихідники коду зберігаються у віддаленому репозиторії (сховищі даних), куди звертаються розробники, щоб забрати актуальну версію файлів або внести зміни. Так вибудовується команда технологія.

2. **log-ування змін у коді.** Відстеження **commit-ів** (внесення змін до коду) допомагає знайти хто, що і коли змінював, вирішити конфлікти при модифікуванні одних і тих же файлів, відкотитися на будь-який попередній стан.

3. **Розгалуження розробки.** Програмісти паралельно ведуть розробку нового функціоналу у окремих гілках, не торкаючись працездатності старого.

4. **Підтримка версії продуктів.** При випуску оновлень програмних продуктів ми позначаємо релізні версії, наприклад, за допомогою **tag-ів**, щоб зафіксувати їх у цьому стані, для дебагу або ретроспективи.

На сьогодні існує розподіл СКВ на три головні категорії:

- **Локальні;**
- **клієнт-серверні (централізовані);**
- **розподілені (децентралізовані).**

В свою чергу СКВ поділяють на **вільні (відкриті)** та **закриті**.

16.3. Локальні системи контролю версій

Кожен користувач як метод контролю версій застосовує копіювання файлів в окремі каталоги, але такий підхід є недосконалим, оскільки не дозволяє відслідковувати всі важливі зміни та відповідно уникнути різник помилок в роботі з файлами, що погіршує якість та продуктивність праці і особливо в складних проєктах.

Для того, щоб вирішити цю проблему, програмісти давно розробили локальні СКВ (рис. 16.1) з простою базою даних, яка зберігає записи про всі зміни в файлах, здійснюючи тим самим контроль ревізій.

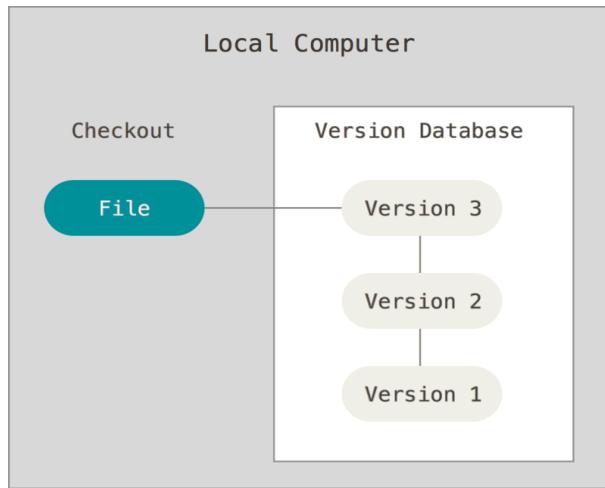


Рис. 16.1 Локальний контроль версій

16.4. Централізовані системи контролю версій

Для вирішення проблеми необхідності тісної взаємодії з іншими розробниками проекту, були розроблені централізовані системи контролю версій (ЦСКВ - рис. 16.2). Такі системи, як **CVS**, **Subversion** і **Perforce**, використовують єдиний сервер, що містить всі версії файлів, і кілька клієнтів, які отримують файли з цього централізованого сховища. Застосування ЦСКВ було стандартом багато років.

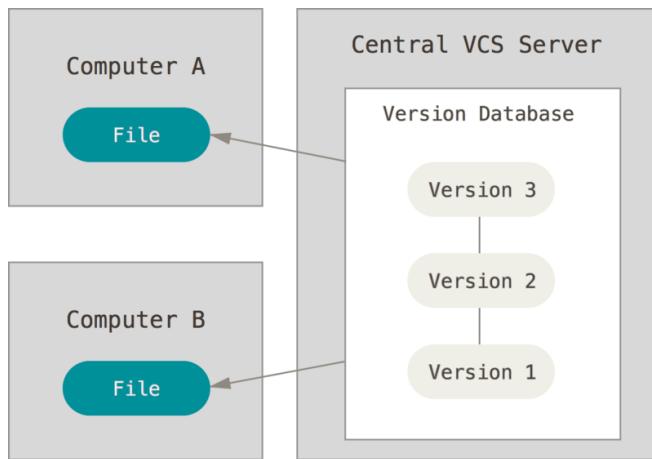


Рис. 16.2 Централізований контроль версій

Централізована СКВ має переваги над іншими СКВ, особливо над локальним. Наприклад, всі розробники проекту певною мірою знають, чим займається кожен із них. Адміністратори мають повний контроль над тим, хто і що може робити і також

набагато простіше адмініструвати ЦСКВ, ніж оперувати локальними базами даних на кожному клієнті.

Мінуси – єдина точка відмови представлена централізованим сервером.

Якщо цей сервер вийде з ладу на годину, протягом цього часу ніхто не зможе використовувати контроль версій для збереження змін, над якими працює, а також ніхто не зможе обмінюватися цими змінами з іншими розробниками. Якщо жорсткий диск, на якому зберігається центральна БД, пошкоджений, а своєчасні бекапи відсутні, ви втратите весь проект разом з історією проекту, не враховуючи одиничних знімків репозиторію, які збереглися на локальних машинах розробників.

Локальні СКВ страждають від тієї самої проблеми: коли вся історія проекту зберігається в одному місці, ви ризикуєте втратити все.

16.5. Розподілені системи контролю версій

Розподілені СКВ (РСКВ - рис. 6.1) вже долають головні недоліки локальних та централізованих СКВ. У РСКВ (таких як **Git**, **Mercurial**, **Bazaar** або **Darcs**) клієнти не просто завантажують знімок всіх файлів (стан файлів на певний момент часу) - вони повністю копіюють репозиторій. У цьому випадку, якщо один із серверів, через який розробники обмінювалися даними, помре, будь-який репозиторій клієнта може бути скопійований на інший сервер для продовження роботи. Кожна копія репозиторію є повним бекапом всіх даних.

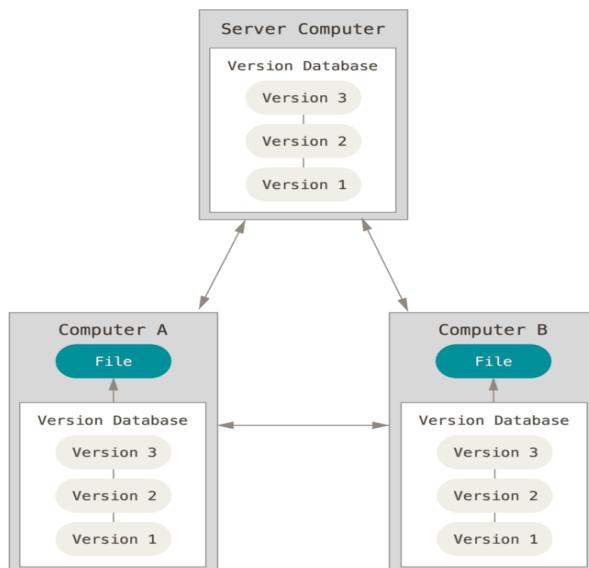


Рис. 16.3 Розподілений контроль версій

Ще однією з переваг РСКВ є те, що більшість з таких систем можуть одночасно взаємодіяти з декількома віддаленими репозиторіями. Завдяки цьому ви можете працювати з різними групами людей, застосовуючи різні підходи одночасно в рамках одного проекту. Це дозволяє застосовувати відразу кілька підходів у розробці, наприклад, ієрархічні моделі, що не можливо в централізованих системах.

16.6. Архітектура Git

Git є розподіленою системою. Центрального репозиторію не існує: всі копії створюються рівними, що різко відрізняється від **VCS** другого покоління, де робота заснована на додаванні та вилученні файлів із центрального репозиторію. Це означає, що розробники можуть обмінюватись змінами один з одним безпосередньо перед об'єднанням своїх змін в офіційну гілку.

Крім того, розробники можуть вносити свої зміни до локальної копії репозиторію без відома інших репозиторіїв. Це дозволяє **commit**-и без підключення до мережі або інтернету. Розробники можуть працювати локально в автономному режимі, доки не будуть готові поділитися своєю роботою з іншими. У цей момент зміни відправляються до інших репозиторіїв для перевірки, тестування або розгортання.

Коли файл додається для відстеження **Git**, він стискається за допомогою алгоритму стиснення **zlib**. Результат хешується за допомогою хеш-функції **SHA-1**. Це дає унікальний хеш, який відповідає конкретному вмісту в цьому файлі. **Git** зберігає його в базі об'єктів, що знаходиться в прихованій папці **.git/objects**. Ім'я файлу – це згенерований хеш, а файл містить стислий контент. Дані файли називаються **blob-ами** і створюються щоразу при додаванні до репозиторію нового файлу (або зміненої версії існуючого файлу).

Git реалізує проміжний індекс (*staging index*), який виступає як проміжна область для змін, які готовуються до **commit**. У міру підготовки нових змін на їхній стислий вміст посилаються в спеціальному індексному файлі, який набуває форми об'єкта дерево. Дерево – це об'єкт **Git**, який пов'язує **blob-и** з їхніми реальними іменами файлів, дозволами на доступ до файлів та посиланнями на інші дерева і таким чином представляє стан певного набору файлів та каталогів. Коли всі відповідні зміни підготовлені для **commit**, індексне дерево можна зафіксувати у репозиторії, який створює об'єкт **commit** у базі даних об'єктів **Git**. **Commit** посилається на дерево заголовків для конкретної версії, а також на автора **commit**, адресу електронної пошти,

дату та повідомлення **commit**. Кожен **commit** також зберігає посилання на свій батьківський **commit**(-и), і так згодом створюється історія розвитку проєкту.

Як уже згадувалося, всі об'єкти **Git** – **blob-и, tree (дерева)** та **commit-и** – стискаються, хешуються та зберігаються в базі даних об'єктів на основі їх хешів. Вони називаються вільними об'єктами (**Loose Objects**). Тут не використовуються ніякі **diff-и** для економії місця, що робить **Git** дуже швидким, оскільки повний вміст кожної версії файлу є вільним об'єктом. Однак деякі операції, такі як передача **commit-ів** у віддалений репозиторій, зберігання дуже великої кількості об'єктів або ручний запуск команди складання сміття **Git** викликають переупаковку об'єктів у пакетні файли. У процесі упаковки обчислюються зворотні **diff-и**, які стискаються для виключення надмірності та зменшення розміру. В результаті створюються файли **.pack** з вмістом об'єкта, а для кожного з них створюється файл **.idx** (або індекс) з посиланням на упаковані об'єкти та їхнє розташування в пакетному файлі.

Коли гілки переміщаються у віддалені сховища або витягуються з них, мережі передаються ці пакетні файли. Під час витягування або вилучення гілок файли пакета розпаковуються для створення вільних об'єктів у репозиторії об'єктів.

Підходи в роботі систем контролю версій

Підхід **Git** до зберігання даних більше схожий на набір знімків мініатюрної файлової системи [15]. Щоразу, коли ви виконуєте параметр «**commit**», тобто зберігаєте стан свого проєкту в **Git**, система запам'ятує, як виглядає кожен файл у цей момент, і зберігає посилання на цей знімок (рис. 16.4). Для збільшення ефективності, якщо файли не були змінені, **Git** не запам'ятує ці файли знову, а лише створює посилання на попередню версію файла, який вже збережений. **Git** представляє свої дані, як, скажімо, потік знімків.

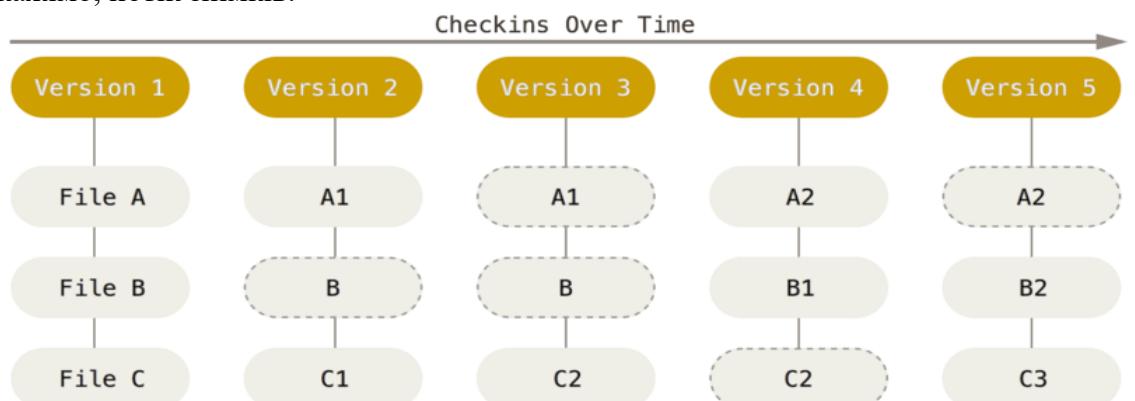


Рис. 16.4 Зберігання даних як знімків проєкту у часі

Це дуже важлива відмінність між **Git** і від будь-якої іншої СКВ. **Git** переосмислює практично всі аспекти контролю версій, скопійованих з попереднього покоління більшістю інших систем. Це робить **Git** більш схожим на мініатюрну файлову систему з напрочуд потужними утилітами, надбудованими над нею, ніж просто на СКВ. Такий підхід надає суттєвих переваг підходам використанням в **Git**.

Виконання операцій локально.

Загальна проблема у роботі більшості СКВ - необхідність постійної підтримки мережевого зв'язку, що призводить до постійних затримок. З метою подолання цієї проблеми більшість операцій **Git** виконує з локальним розташуванням файлів і ресурсів – системі не потрібна жодна інформація з інших комп'ютерів у вашій мережі. Оскільки вся історія проекту зберігається прямо на вашому локальному диску, більшість операцій здається мало не миттевими.

Для прикладу, щоб переглянути історію проекту, **Git** не потрібно з'єднуватися з сервером для її отримання та відображення — система просто читає дані безпосередньо з локальної бази даних. Це означає, що ви побачите історію проекту практично миттєво. Якщо вам необхідно переглянути зміни, зроблені між поточною версією файлу та версією, створеною місяць тому, **Git** може знайти файл місячної давності та локально обчислити зміни, замість того, щоб вимагати віддалений сервер виконати цю операцію, або замість отримання старої версії файлу з сервера та виконання операції локально.

Це також означає, що є лише невелика кількість дій, які ви не зможете виконати, якщо ви знаходитесь офф-лайн або не маєте доступу до VPN на даний момент. Ви можете без будь-яких проблем працювати з вашою локальною копією: коли буде можливість підключитися до мережі, всі зміни можна буде синхронізувати.

Цілісність у **Git** забезпечується обчисленням хеш-суми, і потім відбувається збереження [15]. Всі подальші звернення до збережених об'єктів відбувається за цією хеш-сумою, що включає будь-які випадкові зміни вмісту файлів чи каталогів. Ця функціональність вбудована в **Git** на низькому рівні і є невід'ємною частиною його філософії. Ви не втратите інформацію під час її передачі та не отримаєте пошкоджений файл без відома **Git**.

SHA-1 хеш - механізм, яким користується **Git** для обчислення хеш-сум, становить рядок довжиною 40 шістнадцяткових символів (0-9 і a-f). Він обчислюється на основі вмісту файлу або структури каталогу.

Зразок **SHA-1** хеш:

24b9da6552252987aa493b52f8696cd6d3b00373

Git зберігає всі об'єкти в свою базу даних не за ім'ям, а за хеш-сумою вмісту об'єкта.

Робота **Git** практично заснована на збереженні будь-яких дій додаванням нових даних в базу **Git**. Дуже складно змусити систему видалити дані або зробити щось, що не можна згодом скасувати. Ви можете втратити або зіпсувати свої зміни, поки вони не зафіксовані, але після того, як ви зафіксуєте знімок у **Git**, буде дуже складно щось втратити, особливо, якщо ви регулярно синхронізуете свою базу з іншим репозиторієм.

Вищевказані особливості **Git** розкривають множину переваг, суттєво полегшуючи роботу з проектами, дозволяючи серйозно експериментувати не боячись серйозних проблем.

Три стани Git

У **Git** – файли можуть бути в трьох основних станах – рис. 16.5 [15]:

- змінений (**modified**),
- індексований (**staged**),
- зафіксований (**committed**).

До змінених відносяться файли, які змінилися, але ще не були зафіксовані.

Індексований – це змінений файл у його поточній версії, позначений для включення в наступний «**commit**».

Зафіксований означає, що файл збережений у вашій локальній базі.

Три робочі секції проекту Git

Проект **Git** поділяється на три робочі секції [15]:

- робоча копія (*working tree*),
- область індексування (*staging area*),
- каталог **Git** (*Git directory*).

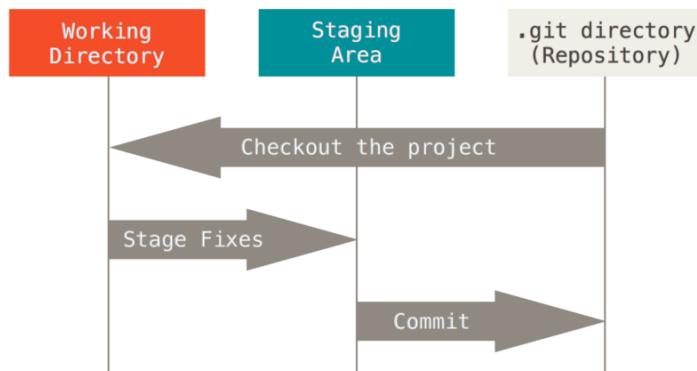


Рис. 16.5 Робоча копія, область індексування та каталог **Git**

Робоча копія є знімком однієї версії проєкту. Ці файли виймаються зі стиснутої бази даних у каталогі **Git** і поміщаються на диск, щоб їх можна було використовувати чи редагувати.

Область індексування - це файл(-и), що зазвичай знаходиться в каталогі **Git**, в ньому міститься інформація про те, що потрапить в наступний «**commit**». Її технічна назва мовою **Git — індекс**, але фраза «**область індексування**» також працює.

Каталог Git - це місце, де **Git** зберігає метадані і базу об'єктів вашого проєкту. Це найважливіша частина **Git** і це частина, яка копіюється при клонуванні репозиторію з іншого комп'ютера.

Базовий підхід у роботі з Git:

1. Ви змінюєте файли вашої робочої копії.
2. Вибірково додаєте до індексу лише ті зміни, які мають потрапити до наступного «**commit**», додаючи тим самим знімки лише цих змін до індексу.
3. Коли ви робите «**commit**», використовуються файли з індексу як є, і цей знімок зберігається у вашому каталогі **Git**.

Якщо певна версія файлу є у каталогі **Git**, то ця версія вважається **зареєстрованою (committed)**. Якщо файл був змінений і доданий до індексу, значить, він **індексований (staged)**. І якщо файл був змінений з моменту останнього розпакування з репозиторію, але не був доданий до індексу, він вважається **зміненим (modified)**.

16.7. Директорія Git та робоча директорія

В “директорії **git**” зберігається вся історія **Git** та мета-інформація вашого проєкту - включаючи всі об'єкти (**commit**, дерева, **blob**-и, **tag**-и), всі покажчики на різні гілки та багато іншого [16].

На кожен проєкт є тільки одна директорія **Git**, і це директорія (за замовчуванням, але не обов'язково) “**.git**” в корені вашого проєкту. Якщо ви подивитеся на вміст цієї директорії, то побачите всі ваші важливі файли (можуть бути і інші файли/директорії):

```
$>tree -L 1  
.  
/-- HEAD      # вказівник на вашу активну гілку  
/-- config    # ваші персональні налаштування  
/-- description # опис проєкту
```

```
-- hooks/      # pre/post action hooks (скрипти (надалі хуки) які можуть викликатися git командами)
-- index      # індексний файл
-- logs/       # історія гілок проекту (де вони розташовані)
-- objects/    # ваші об'єкти (commit, дерева, blob-u, tag-u)
`-- refs/     # вказівники на ваші гілки розробки
```

Робоча директорія це просто тимчасове місце, де ви можете модифікувати файли, а потім виконати **commit**.

16.8. Об'єктна модель Git

Історія проекту зберігається у особливо організованих файлах, що посилаються один на одного за допомогою 40-значного "імені об'єкта" **SHA1** – криптографічної хеш-функції [17].

Це дозволяє:

- **Git** може швидко визначити чи ідентичні два об'єкти чи ні, просто порівнюючи їх імена.
- Так як імена об'єктів обчислюються однаково у всіх репозиторіях, то об'єкти з однимаковим вмістом у двох репозиторіях завжди зберігаються під одинаковими іменами.
- **Git** може знаходити помилки, коли читає об'єкт, для цього потрібно просто порівняти хеш значення вмісту об'єкта з його ім'ям.

Об'єкти

Основою побудови **Git** є чотири типи об'єктів - "**блоб**" (**blob**), "**дерево**" (**tree**), "**commit**", та "**таг**" (**tag**). Кожен об'єкт має три частини – тип, розмір та зміст. Таку структуру порівнюють з надбудовою над традиційною файловою системою.

"**blob**" використовується щоб зберігати вміст файлу - зазвичай це просто файл.

"**tree**" це щось на зразок директорії - воно посилається на групу інших дерев та/або **blob**-ів (тобто файлів та директорій).

"**Commit**" вказує на окреме дерево, за суттю відзначає дерево фіксуючи в історії яким чином воно виглядає в момент виконання **commit**. Він містить мета-інформацію фіксуючи момент часу та автора змін, внесених з останнього **commit**, покажчик на попередній **commit** тощо.

"**tag**" це спосіб маркувати певним чином певний **commit**. Зазвичай це використовується щоб маркувати (по суті дати якесь ім'я, що легко запам'ятовується) певні **commit** є специфічними, щоб згодом було легше їх знайти.

Об'єкт типу «Блоб» (blob).

Blob зберігає зміст файлу (рис. 16.6).

5b1d3..	
blob	size
<pre>#ifndef REVISION_H #define REVISION_H #include "parse-options.h" #define SEEN (1u<<0) #define UNINTERESTING (1u #define TREESAME (1u<<2)</pre>	

Рис. 16.6 Представлення об'єкту типу Блоб

Об'єкт "**blob**" це порція бінарних даних. **Blob** ні на що не посилається у нього немає жодних атрибутів, не має навіть імені файла.

Оскільки **blob** повністю визначається його власним вмістом, то якщо два файли в директорії або навіть у різних версіях репозиторію мають одинаковий вміст, вони будуть розділяти той самий **blob** об'єкт. Об'єкт повністю залежить від розташування в дереві каталогів, і перейменування файла не змінить об'єкт із яким файл пов'язаний.

Зміст **blob** можливо переглянути командою **git show**. Наприклад:

```
$ git show 6ff87c4664
```

*Note that the only valid version of the GPL as far as this project
is concerned is _this_ particular version of the license (ie v2, not
v2.2 or v3.x or whatever), unless explicitly otherwise stated.*

...

Об'єкт «Дерево» (Tree)

Tree це простий об'єкт який містить у собі групу покажчиків на **blob**-и та інші дерева (рис. 16.7) - представляє вміст директорії чи піддиректорії.

c36d4..		
tree	size	
blob	5b1d3	README
tree	03e78	lib
tree	cdc8b	test
blob	cba0a	test.rb
blob	911e7	xdiff

Рис. 16.7 Представлення об'єкту дерево

Щоб оглянути зміст дерева необхідно використати команду **git ls-tree** (можливо використовувати **git show**, але вона надає менший обсяг інформації про зміст дерева). Наприклад за відомим значенням **SHA** дерева:

```
$ git ls-tree fb3a8bdd0ce
100644 blob 63c918c667fa005ff12ad89437f2fdc80926e21c .gitignore
100644 blob 5529b198e8d14decbe4ad99db3f7fb632de0439d .mailmap
100644 blob 6ff87c4664981e4397625791c8ea3bbb5f2279a3 COPYING
040000 tree 2fb783e477100ce076f6bf57e4a6f026013dc745 Documentation
100755 blob 3c0032cec592a765692234f1cba47dfdcc3a9200 GIT-VERSION-GEN
100644 blob 289b046a443c0647624607d471289b2c7dcd470b INSTALL
100644 blob 4eb463797adc693dc168b926b6932ff53f17d0b1 Makefile
100644 blob 548142c327a6790ff8821d67c2ee1eff7a656b52 README
...
...
```

Об'єкт **tree** містить список записів, що складаються з виду, типу об'єкту, значення **SHA1**, і імені відповідно. Записи відсортовані на ім'я. Так виглядає вміст однієї директорії **tree**.

Посилання на об'єкт у дереві може бути як **blob** (файлом по суті), так і **tree** (піддиректорією). Оскільки імена всіх об'єктів, **tree** та **blob**, збігаються з **SHA** хеш-значенням їхнього вмісту, то **SHA** значення двох **tree** будуть ідентичні тільки якщо їх вміст (включаючи, рекурсивно, вміст усіх піддиректорій) ідентичний.

Об'єкт commit

Об'єкт "**commit**" пов'язує фізичний стан **tree** з описом того, яким чином ми прийшли до нього і чому (рис. 16.8).

ae668..	
commit	size
tree	c4ec5
parent	a149e
author	Scott
committer	Scott
my commit message goes here and it is really, really cool	

Рис. 16.8 Представлення об'єкту **commit**

Щоб дослідити **commit** можна застосувати параметр **--pretty=raw** з **git show** або **git log**. Наприклад:

```
$ git show -s --pretty=raw 2be7fcb476
```

```
commit 2be7fcb4764f2dbce52635b91fdb1b3dcf7ab4
tree fb3a8bdd0ceddd019615af4d57a53f43d8cee2bf
parent 257a84d9d02e90447b149af58b271c19405edb6a
author Dave Watson <dwatson@mimvista.com> 1187576872 -0400
committer Junio C Hamano <gitster@pobox.com> 1187591163 -0700

    Fix misspelling of 'suppress' in docs

    Signed-off-by: Junio C Hamano <gitster@pobox.com>
```

Commit визначається наступними головними значеннями:

- **Дерево (tree)**: ім'я **SHA1** об'єкта **tree** (як визначено нижче), що представляє вміст директорії в певний момент часу.
 - **Батько(и)**: **SHA1** ім'я деякого числа **commit**, які являють собою попередній крок(и) в історії проекту. Приклад вище має одного з батьків; хоча **commit**-и злиття можуть мати більше одного батька. **Commit** без батьків називається "**root (кореневий) commit**", і є початковим станом проекту. Кожен проект повинен мати принаймні один кореневий **commit**. Проект може також мати багато коренів, проте це не загальний випадок (і не обов'язково хороша ідея).
 - **Автор**: Ім'я розробника відповідального за ці зміни, разом із датою.
 - **Commit-ер**: ім'я розробника, який створив цей **commit**, разом з датою цієї події. Воно (ім'я) може відрізнятись від імені автора; наприклад у випадку, якщо автор написав патч і відправив його електронною поштою іншому розробнику який наклав патч і виконав **commit**.
 - **Коментар**: описує цей **commit**.

Об'єктна модель

Розглянемо приклад поєднання 3х вище розглянутих головних об'єкті (**blob**, **tree** та **commit**).

Для проекту директорії зразкової структури (рис. 16.9):

```
$>tree
.
|-- README
`-- lib
    |-- inc
    |   '-- tricks.rb
    '-- mylib.rb

2 directories, 3 files
```

Рис. 16.9 Директорія зразкової структури

За умови, якщо виконано **commit** в репозиторій **Git** отримаєм наступну відображення (рис. 16.10).

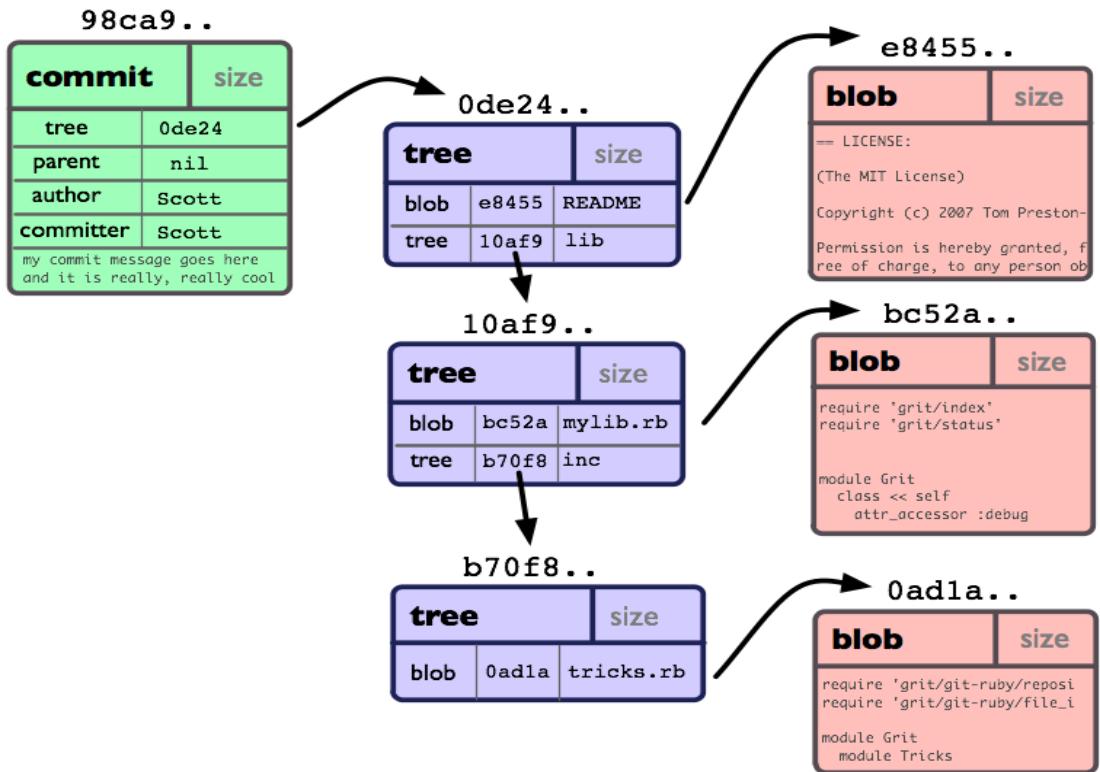


Рис. 16.10 Представлення зразка директорії після **commit**

tag	size
object ae668	
type commit	
tagger Scott	
my tag message that explains this tag	

Рис. 16.11 Представлення об'єкту таг

Об'єкт «tag» (tag)

Об'єкт **tag** містить **ім'я об'єкту** (називається просто - “об'єкт”), **тип об'єкту**, **ім'я tag**, або розробника (“таггер”) який створив **tag**, і **повідомлення**, яке може

містити підпис (рис. 16.11). Оглянути об'єкт **tag** можна командою **git cat-file**. Наприклад:

```
$ git cat-file tag v1.5.0
object 437b1b20df4b356c9342dac8d38849f24ef44f27
type commit
tag v1.5.0
tagger Junio C Haemano <junkio@cox.net> 1171411200 +0000

GIT 1.5.0
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.6 (GNU/Linux)

iD8DBQBF0lGqwMbZpPMRm5oRAuRiAJ9ohBLd7s2kqjkKlqIqqC57SbnmzQCdG4
ui
nLE/L9aUXdWeTFPron96DLA=
=2E+0
-----END PGP SIGNATURE-----
```

Питання до розділу 16

1. Що таке системи контролю версій?
2. Які задачі виконують системи контролю версій?
3. Які категорії систем контролю версій ви знаєте? Опишіть їх.
4. Які покоління систем контролю версій ви знаєте? Опишіть їх.
5. Що ви знаєте про третє покоління систем контролю версій?
6. Як побудована Git?
7. Які підходи в побудові роботи СКВ ви знаєте? Яка відмінність Git від інших СКВ?
8. Як забезпечується цілісність Git?
9. Як організована директорія Git?
10. Об'єктна модель Git? Опишіть її.
11. Опишіть об'єкт blob.
12. Опишіть об'єкт tree.
13. Опишіть об'єкт commit.
14. Опишіть об'єкт tag.

РОЗДІЛ 17. УПРАВЛІННЯ ІТ ПРОЄКТОМ В GITHUB.

17.1. Управління ІТ-проєктом в GitHub

Типова загальна послідовність ведення проєкту ІТ в GitHub:

1. Реєстрація особистого акаунту GitHub.
2. Створення організації.
3. Створення команд, що належать до організації та запрошення учасників як в команду так і вільних учасників проєкту, що не належать до команд. Призначення ролей учасникам.
4. Поділ проєкту на функції та розподіл їх виконання між командами та учасниками проєкту. Вибір моделі контролю версій ведення проєкту – магістральний метод чи Git-flow.
5. Відповідно до вимог ІТ проєкту в організації створюються репозиторії. В GitHub, в залежності від специфіки проєкту, репозиторії створюють, як окремі проєкти, або як окремі функції проєкту.
6. До співпраці над окремими частинами проєкту (окремі репозиторії) також можливо створювати команди та запрошувати окремих вільних працівників (поза командою). Призначення ролей учасників.
7. Робота з гілками проєкту.

Створюють основні гілки **main** та **develop**. Гілку **main** використовують як гілку в яку подаються лише закінчені релізні версії програмного продукту. В процесі роботи над проєктом до гілки розробки (**develop**) добавляють (**комміти**) гілки окремих функцій проєкту. По завершенню робіт над функціями чи проєктом у процесі доведення до релізного варіанту проєкту виникають гілки виправлення, що враховують зміни до початкового варіанту релізу програмного проєкту чи функцій. Узгоджений, перевірений та протестований варіант з гілки **develop** commit-ом подають до релізної гілки **main**.

8. Складні ситуації злиття різних гілок внаслідок праці над ними різних команд та учасників проєкту вирішуються злиттям за узгодженням та під контролем учасників проєкту покроково.

17.2. Початкові налаштування Git на локальному пристрой

До складу **Git** входить утиліта **git config**, яка дозволяє переглядати та налаштовувати параметри, що контролюють усі аспекти роботи **Git**, а також його зовнішній вигляд. Ці параметри можуть бути збережені у трьох місцях [15]:

1. Файл *[path]/etc/gitconfig* містить значення, загальні для всіх користувачів системи та всіх їх репозиторіїв. Якщо при запуску *git config* вказати параметр *--system*, параметри читаються і зберігаються саме в цей файл. Так як цей файл є системним, то вам потрібні права супер-користувача для внесення змін до нього.

2. Файл *~/.gitconfig* або *~/.config/git/config* зберігає установки конкретного користувача. Цей файл використовується при вказівці параметра *--global* і застосовується до всіх репозиторіїв, з якими ви працюєте у поточній системі.

3. Файл *config* у каталозі **Git** (тобто *.git/config*) репозиторію, який ви використовуєте в даний момент, зберігає налаштування конкретного репозиторію. Ви можете змусити **Git** читати і писати в цей файл за допомогою *--local*, але насправді це значення за замовчуванням. Не дивно, що вам потрібно бути десь у репозиторії **Git**, щоб ця опція працювала правильно.

Налаштування на кожному наступному рівні підміняють налаштування з попередніх рівнів, тобто значення *.git/config* перекривають відповідні значення *[path]/etc/gitconfig*.

У системах сімейства **Windows Git** шукає файл *.gitconfig* у каталозі **\$HOME** (**C:\Users\\$USER** для більшості користувачів). Крім того, **Git** шукає файл *[path]/etc/gitconfig*, але вже щодо кореневого каталогу **MSys**, який знаходиться там, куди ви вирішили встановити **Git** під час запуску інсталятора.

Якщо ви використовуєте **Git** для **Windows версії 2.x або новіше**, то також обробляється файл конфігурації рівня системи, який має шлях **C:\Documents and Settings\All Users\Application Data\Git\config** у **Windows XP** або **C:\ProgramData\Git\config** у **Windows Vista** та новіший. Цей файл може бути змінений лише за допомогою команди *git config -f <file>*, запущеної з правами адміністратора.

Щоб переглянути всі встановлені налаштування та дізнатися, де саме вони задані, використовуйте команду:

```
$ git config --list --show-origin
```

Дані користувача:

Для того, щоб приступити до роботи необхідно ідентифікувати користувача (опція *-global* застосовує налаштування для всіх операцій в системі):

```
$ git config --global user.name «John Doe»  
$ git config --global user.email «johndoe@example.com»
```

Вибір редактору:

```
$ git config --global core.editor emacs
```

Налаштування гілки за замовчуванням:

```
$ git config --global init.defaultBranch main
```

Перевірка налаштувань:

```
git config --list
```

17.3. Організація. команда. репозиторій. ролі

Організаційною основою роботи з GitHub є “Організації” в просторі яких знаходяться їх проекти. Облікові записи організацій представляють групи людей з сумісним правом власності проектів та різними інструментами для керування групами.

Щоб створити організацію необхідно обрати іконку “+” нагорі праворуч на будь-якій сторінці GitHub та виберіть у меню “New organization” (нова організація) (рис. 17.1).

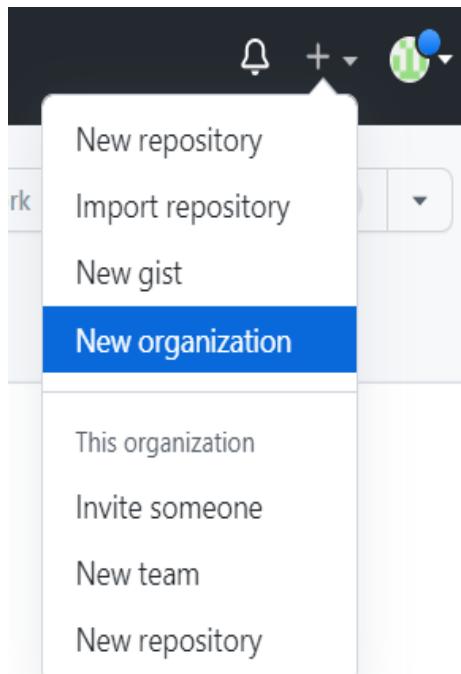


Рис. 17.1 Створення нової організації

Переходимо до сторінки вибору (рис. 17.2).

The screenshot shows the 'Choose a plan' section of the GitHub website. At the top, it says 'Pick a plan for your organization'. Below that, it asks 'How often do you want to pay?' with options for 'Monthly' (selected), 'Yearly', and 'Get 1 month free'. There are three main plan cards:

- Free**: \$0 per year forever. Includes unlimited public/private repositories, automatic security and version updates, 2,000 CI/CD minutes/month for public repositories, 500MB of Packages storage for public repositories, and new issues & projects (in limited beta). A 'Create a free organization' button is available.
- Team**: \$48 per user/year for the first 12 months*. Includes everything in Free plus... (Access to GitHub Codespaces, Protected branches, Multiple reviewers in pull requests, Draft pull requests), plus 2,000 CI/CD minutes/month for public repositories, 500MB of Packages storage for public repositories, and new issues & projects. A 'Continue with Team' button is available.
- Enterprise**: \$252 per user/year for the first 12 months*. Includes everything in Team plus... (Enterprise Managed Users, User provisioning through SCIM, Enterprise Account to centrally manage multiple organizations, Environment protection rules). A 'Start a free trial' and 'Contact Sales' button is available.

Рис. 17.2 Вибір плану для вашої організації

Де обираємо план “Free” і потім обираємо назву організації та контактну адресу електронної пошти для організації (рис. 17.3).

The screenshot shows the 'Set up your organization' page. It has a header 'Tell us about your organization' and a title 'Set up your organization'. The form fields are:

- Organization account name ***: An input field with placeholder text 'Organization account name'.
- Contact email ***: An input field with placeholder text 'Contact email'.
- This organization belongs to: ***: A dropdown menu with two options:
 - My personal account
I.e., GitTEF-APEPS
 - A business or institution
For example: GitHub, Inc., Example Institute, American Red Cross
- Verify your account**: A large input field containing a large green checkmark.
- Acceptance checkbox**: A checkbox with the text 'I hereby accept the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement.'

Рис. 17.3 Сторінка початкових налаштувань організації

Далі обираєте варіант вашого аккаунта – чи персональний, чи бізнес організації. Проводите верифікацію та підтверджуєте згоду і обираєте продовжити (Next).

Наступною сторінкою буде сторінка завершення створення організації, що пропонує пошуком додати учасників до організації шляхом пошуку ім'я користувача, повного ім'я або адреси електронної пошти (рис. 17.4).

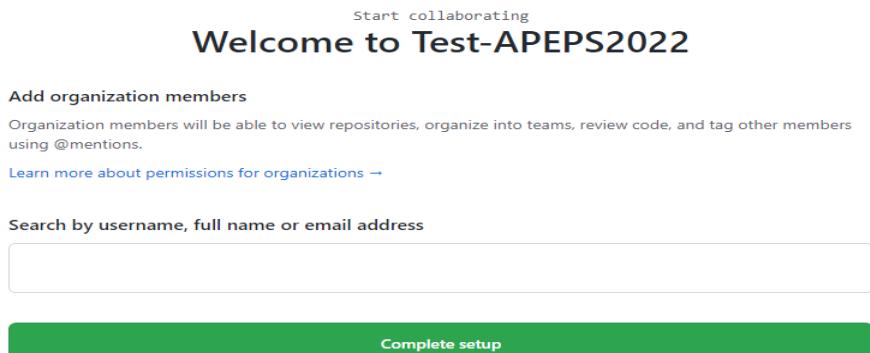


Рис. 17.4 Сторінка додання членів до організації

Наступна сторінка є опитувальною (рис. 17.5).

The screenshot shows the 'Welcome to GitHub' survey page. It starts with a message: 'Woohoo! You've joined millions of developers who are doing their best work on GitHub. Tell us what you're interested in. We'll help you get there.' Below this is a section titled 'Tell us about you' with the heading 'What do you spend time on most day-to-day?'. It asks users to select all that apply from a list of activities: 'Writing code', 'Managing and coordinating engineering work', 'Planning projects', and 'Billing administration'. There's also an 'Other' section with a text input field 'Please describe'. Further down, there's another section titled 'Tell us about your team' with the heading 'How many people do you expect to actively work within this GitHub organization?'. It features a row of five radio buttons for selecting the number of people: '0', '1-5', '6-15', '16-24', and '25+'.

Рис. 17.5 Сторінка “Welcome to GitHub”

Використовуючи в правому верхньому куті значок вашого профілю ви можете переглянути інформацію за вашою організацією (рис. 17.6). Де ви отримаєте доступ до різних даних вашої організації.

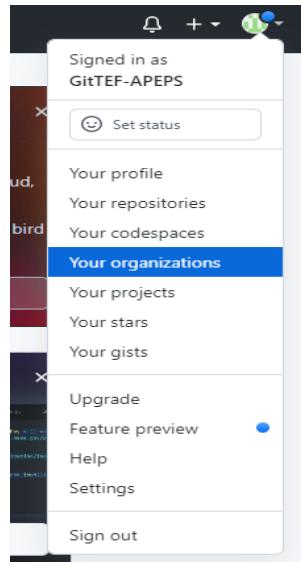


Рис. 17.6 Перехід до сторінки інформації про організацію

Також доступ до можливості управлія організаціями розташовано в лівому верхньому куті (рис. 17.7).

A screenshot of the GitHub personal dashboard for the user 'GitTEF-APEPS'. On the left, there is a sidebar with options: Switch dashboard context (GitTEF-APEPS is selected), Manage organizations, and Create organization. Below this is a Recent activity section and a Your teams section. The main area is titled 'The home for all developers — includir' and contains sections for creating a new repository (with a 'Create a new repository' button), using tools of the trade (with a 'Simplify your development workflow with a GUI' button), and installing a GitHub desktop application. A sidebar on the right shows a list of organizations: GitTEF-APEPS, Software-Engineering-in-Energy, Test-APEPS2022, and others.

Рис. 17.7 Управління організаціями

Як і особисті облікові записи, організації безкоштовні, якщо все, що ви будете в них зберігати буде відкритим кодом.

Як власник організації, коли ви робите **fork** сховища, у вас буде вибір: робити **fork** у вашому власному просторі імен, чи у просторі імен організації. Коли ви створюєте нові сховища, ви можете створити їх або під особистим обліковим записом, або під будь-якою організацією, що її власником є ви. Також ви автоматично будете “слідкувати” (**watch**) за всіма сховищами, що ви створили для цих організацій.

Ви маєте головну сторінку організації, на якій є список усіх ваших сховищ — її можуть бачити й інші люди.

Команди

Організації можуть працювати, як з окремими людьми так і з окремими людьми через команди.

Щоб керувати командами потрібно на сторінці організації обрати і перейти за “Teams” (рис. 17.8).

The screenshot shows the GitHub organization page for 'Software-Engineering-in-Energy'. At the top, there's a navigation bar with links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the navigation bar, the organization's logo is displayed, along with the name 'Software-Engineering-in-Energy', the number of followers (2), and the location (Ukraine). A 'Follow' button is also present. The main content area has tabs for 'Overview', 'Repositories', 'Projects', 'Packages', 'Teams', 'People', and 'Settings'. The 'Teams' tab is currently selected. On the left, there's a sidebar titled 'Repositories' with a search bar and filters for 'Type', 'Language', and 'Sort'. It lists several repositories: 'Olena_Patsevko' (Java, 0 stars, 0 forks, 0 issues, updated 2 hours ago), 'Anton-Makarenko' (Java, 0 stars, 0 forks, 0 issues, updated 3 days ago), 'Test' (Private, 0 stars, 0 forks, 0 issues, updated 3 days ago), 'IhorNekhaienko' (Python, 0 stars, 0 forks, 0 issues, updated on 18 Jul), and 'SerhiiiSlipchenko' (Python, 0 stars, 0 forks, 0 issues, updated on 28 Jun). To the right of the repository list, there's a section titled 'View as: Public' with a note about viewing the README and pinned repositories as a public user. It also includes a link to 'Get started with tasks that most successful organizations complete'. Further down, there's a 'People' section showing profile icons for team members and a 'Top languages' section indicating Python and Java.

Рис. 17.8 Керування командами

Ви опинитесь на сторінці, що буде відображати команди організації та можливість створити нові – “New team” (рис. 17.9).

Software-Engineering-in-Energy

Overview Repositories 7 Projects Packages Teams 2 People 11 Settings

Seamless communication with teams

Teams are a great way for groups of people to communicate and work on code together. Take a look at why they're great.

- Flexible repository access**
You can add repositories to your teams with more flexible levels of access (Admin, Write, Read).
- Request to join teams**
Members can quickly request to join any team. An owner or team maintainer can approve the request.
- Team mentions**
Use team @mentions (ex. @github/design for the entire team) in any comment, issue, or pull request.

[Learn more](#)

Find a team... [New team](#)

	Visibility ▾	Members ▾
<input type="checkbox"/> Select all		
<input type="checkbox"/> Extracting knowledge from content ... China Project		5 members 1 team ▾
<input type="checkbox"/> project with china 2022 Scenario modeling in data analysis		3 members 0 teams

Рис. 17.9 Сторінка “Teams”

З’являється можливість приєднатися до дискусії стосовно команди (рис. 17.10).

Select all

Extracting knowledge from content ...
China Project

project with china 2022
Scenario modeling in data analysis

Рис. 17.10 Перехід до дискусій команд

або переглянути профілі членів команди, або створити команду – “New team” (рис. 17.11).

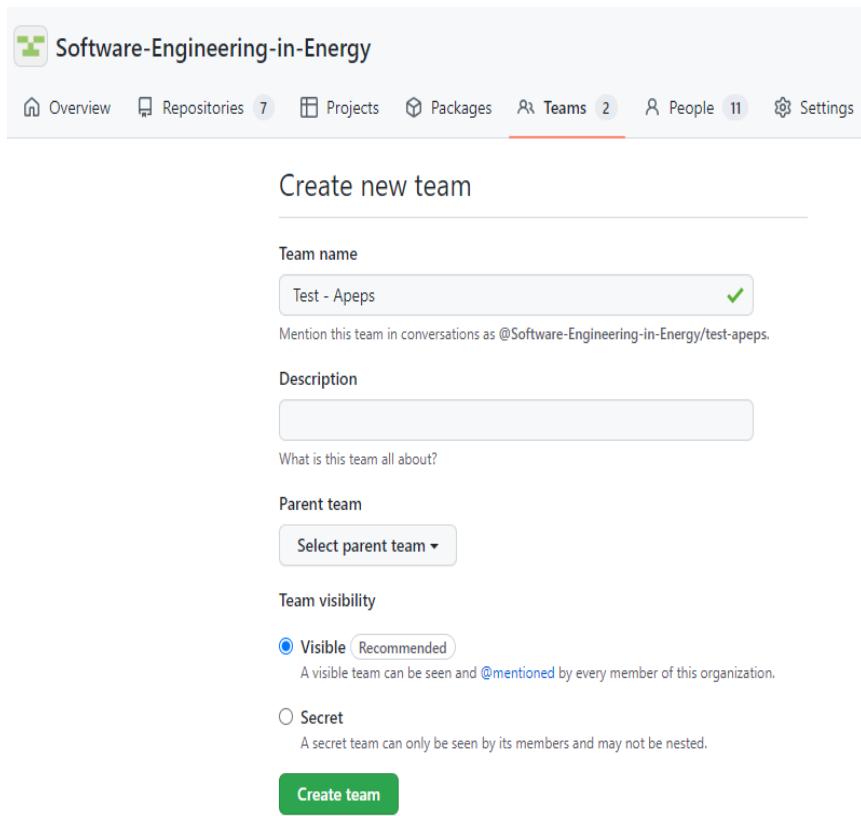


Рис. 17.11 Створення команди

Наступна сторінка після реєстрації команди відкриває можливість вести дискусію та з лівої сторони має можливість додавати “+” користувачів (рис. 17.12).

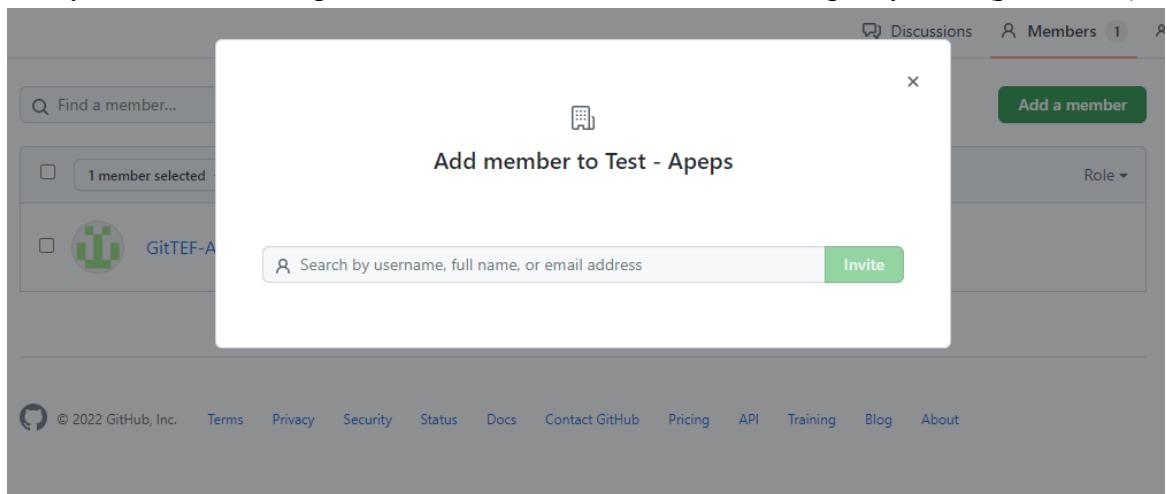


Рис. 17.12 Додавання користувачів в команди

Коли ви когось запрошуєте до команди, вони отримають листа, що повідомить їм про запрошення.

Створення репозиторію

На головній сторінці в лівій частині є можливість обрати репозиторій, або

створити новий обравши  **New** (рис. 17.3).

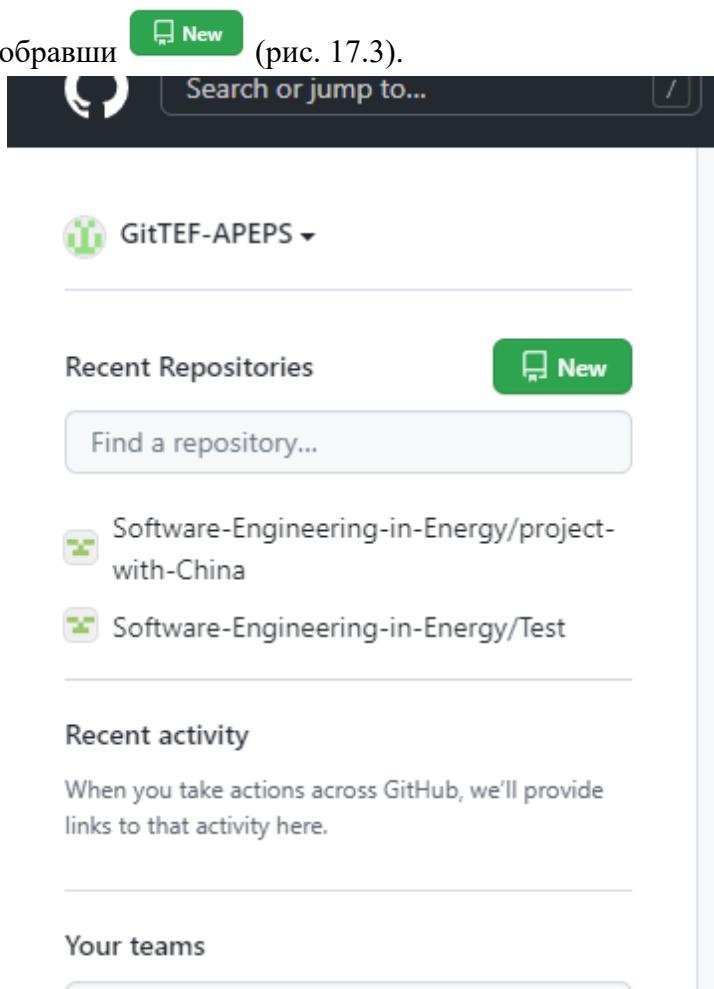


Рис. 17.13 Створення репозиторію

На новій сторінці відкривається можливість обрати показники нового репозиторію (рис. 17.4) – назvu, опис (не обов'язковий), публічний чи приватний, README file, **.gitignore** (які файли будуть проігноровані при **commit**), чи є ліцензія та створити репозиторій.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *



Repository name *

/

Great repository names are short and memorable. Need inspiration? How about [turbo-bassoon?](#)

Description (optional)

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▾

You are creating a public repository in your personal account.

Create repository

Рис. 17.14 Сторінка створення репозиторію

Ролі

Якщо увійти до організації то налаштування - , дають змогу перейти до сторінки де ви можете обрати (ролі в репозиторії) і ознайомитись з ролями (рис. 17.15), що призначені за замовчуванням чи створити нові ролі за вашими вимогами.

Repository roles

Roles are used to grant access and permissions for teams and members. In addition to the available pre-defined roles, you can create up to 0 custom roles to fit your needs. [Learn more](#)

Pre-defined roles	
<input type="checkbox"/> Read	Read and clone repositories. Open and comment on issues and pull requests.
<input type="checkbox"/> Triage	Read permissions plus manage issues and pull requests.
<input type="checkbox"/> Write	Write permissions plus read, clone and push to repositories.
<input type="checkbox"/> Maintain	Write permissions plus manage issues, pull requests and some repository settings.
<input type="checkbox"/> Admin	Full access to repositories including sensitive and destructive actions. Modify Admin Role

Custom roles

[Create a role](#)

Create custom roles with GitHub Enterprise

Enterprise accounts offer organizations more granular control over permissions by allowing you to configure up to three custom repository roles. This enables greater control over who and how your users access code and data in your organization.

Рис. 17.15 Ролі в репозиторії

Read - Ви можете читати та клонувати сховища, відкривати та коментувати проблеми і запити на вилучення.

Triage – Ви маєте дозвіл на читання та керувати проблемами та запитами на отримання.

Write – Ви можете сортувати дозволи, а також читати, клонувати та надсилати до репозиторіїв.

Maintain – Ви маєте дозволи на запис, а також керування проблемами, запити на отримання та деякі налаштування сховища.

Admin – Ви маєте повний доступ до репозиторіїв включаючи конфіденційні та деструктивні дії. Також ви маєте можливість відредактувати роль **Admin**.

Якщо ви зайдете на сторінку певного репозиторію в організації то шляхом вибору налаштувань (рис. 17.16).



Рис. 17.16 Налаштування

ви маєте можливість переглянути учасників обравши (рис. 17.17).

Collaborators and teams

Рис. 17.17 Співробітники та команди

і опинитесь на сторінці, де можливо обрати ролі для кожного учасника (рис. 17.18 – 17.19).

The screenshot shows the GitHub repository settings page for 'Software-Engineering-in-Energy / project-with-China'. The 'General' tab is selected. On the left, there's a sidebar with sections like 'Access', 'Code and automation', 'Pages', 'Security', and 'Integrations'. The 'Collaborators and teams' section is highlighted under 'Access'. On the right, the main area shows the repository name 'project-with-China' and options to 'Rename' or 'Template repository'. Below that, there are checkboxes for 'Require contributors to sign off on web-based commits' and 'Social preview' (with a note about uploading a social image). At the bottom, there's a 'Download template' button.

Рис. 17.18 Перехід до перегляду учасників репозиторію

Рис. 17.19 Управління доступом учасникам репозиторію

Read - Ви можете читати та клонувати це сховище, відкривати та коментувати проблеми і запити на вилучення.

Triage (Сортування) – Ви можете читати та клонувати це сховище. Також ви можете керувати проблемами та витягувати запити.

Write – Ви можете читати, клонувати та надсилати в це сховище. Також можете керувати проблемами та витягувати запити.

Maintain – Ви можете читати, клонувати та надсилати в це сховище. Також ви можете керувати проблемами, запитами на отримання та деякими налаштуваннями сховища.

Admin – Ви можете читати, клонувати та надсилати в це сховище. Також можете керувати проблемами, запитами на отримання та налаштування сховища, зокрема додавати співавтора.

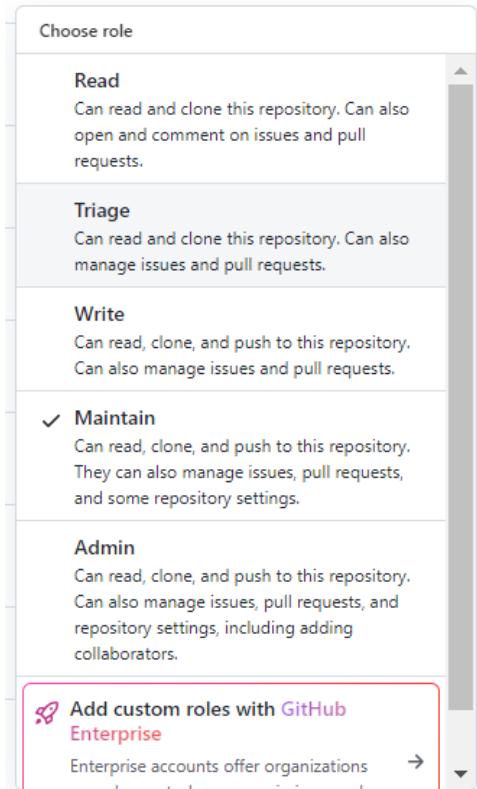


Рис. 17.20 Ролі в репозиторії

Більш детально як власники організацій можуть назначати ролі окремим особам та групам (рис. 17.20), надаючи їм різні набори дозволів в організації та як налаштовувати доступ до кожного репозиторію в вашій організації, призначити детальовані ролі, надаючи людям доступ до необхідних їм функцій та задачам можна ознайомитись в наступних джерелах [18, 19].

Питання до розділу 17

1. Що таке Git та GitHub?
2. Процедура реєстрації GitHub та встановлення Git?
3. Початкові налаштування Git на локальному пристрої?
4. Налаштування за замовчуванням?
5. Організаційна основа GitHub?
6. GitHub. Опишіть організації, команди, репозиторії?
7. GitHub. Ролі в організації?
8. GitHub. Ролі в репозиторії?

РОЗДІЛ 18. КРОКИ РЕАЛІЗАЦІЇ ІТ-ПРОЄКТІВ В GITHUB

18.1. Фіксація в Git

Можливості проектів розширяють розгалуження, що підтримують більшість СКВ. **Git** заохочує процес роботи, при якому розгалуження та злиття виконується часто: операція створення гілки виконується майже миттєво, перемикання між гілками, як правило, також швидко. Розгалуження, як функціональність надає унікальний та потужний інструмент, який може суттєво підвищити ефективність змінити звичний процес розробки [15].

Раніше було розглянуто, що **Git** зберігає дані як послідовність знімків.

Об'єкт фіксації **Git** зберігає **вказівник на знімок змісту**, який ви додали. Цей об'єкт також містить:

- ім'я,
- поштову адресу автора,
- набране вами повідомлення,
- вказівники на фіксацію або фіксації, що були прямо до цієї фіксації (батько чи батьки): нуль для першої фіксації, одна фіксація для нормальній фіксації, та декілька фіксацій для фіксацій, що вони є результатом злиття двох чи більше гілок.

Приклад: припустимо, що у вас є тека з трьома файлами, які ви додали та зафіксували. Додання файлів обчислює контрольну суму для кожного (**SHA-1** хеш про котрій ми згадували раніше), зберігає версію файлу в сховищі **Git** (**Git** називає їх **blob-ами**), та додає їх контрольні суми до області додавання:

```
$ git add README test.rb LICENSE  
$ git commit -m 'The initial commit of my project'
```

Коли ви створили фіксацію за допомогою **git commit** (рис. 18.1), **Git** обчислив контрольну суму кожної теки (у цьому випадку, тільки кореневої теки) та зберігає ці об'єкти дерева в сховищі **Git**. Потім **Git** створює об'єкт фіксації, що зберігає метадані та вказівник на корінь дерева проекту, щоб він міг відтворити цей знімок, коли потрібно.

Ваше **Git** сховище тепер зберігає п'ять об'єктів: по одному **blob** зі змістом на кожен з трьох файлів, одне **tree**, що перелічує зміст теки та вказує, які файли зберігаються у яких **blob**, та одну фіксацію, що вказує на корінь дерева, та зберігає метадані фіксації.

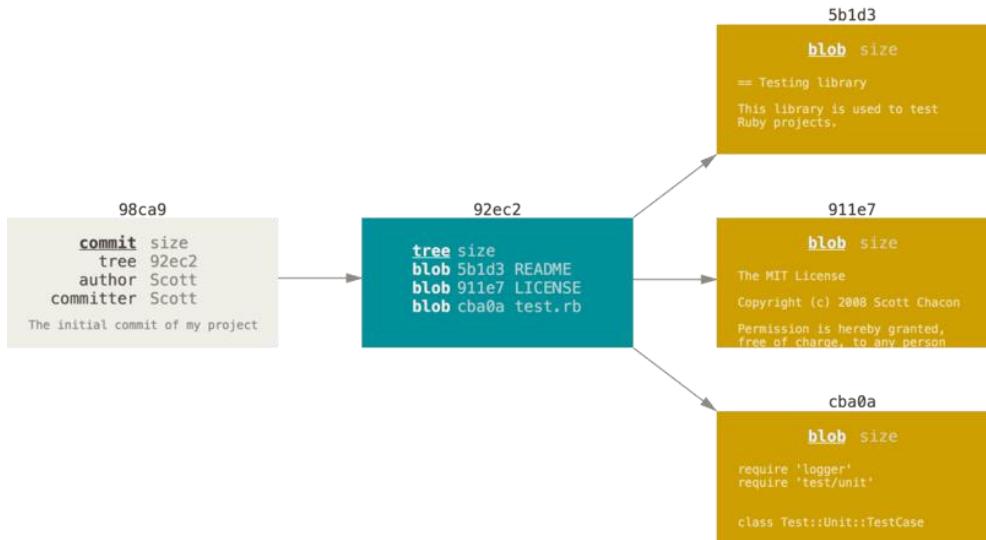


Рис. 18.1 Фіксація як дерево

Ваше **Git** сховище тепер зберігає п'ять об'єктів: по одному **blob** зі змістом на кожен з трьох файлів, одне **tree**, що перелічує зміст теки та вказує, які файли зберігаються у яких **blob**, та одну фіксацію, що вказує на корінь дерева, та зберігає метадані фіксації.

Якщо ви зробите якісь зміни та зафіксуєте знову, наступна фіксація буде зберігати вказівник на попередню.

Гілка в **Git** це просто легкий вказівник, що може пересуватись, на одну з цих фіксацій. Загальноприйнятим ім'ям першої гілки в **Git** є **master**. Коли ви почнете робити фіксації, вам надається гілка **master**, що вказує на останню зроблену фіксацію. Щоразу, коли ви фіксуєте, вона переміщується вперед автоматично (рис. 18.2).

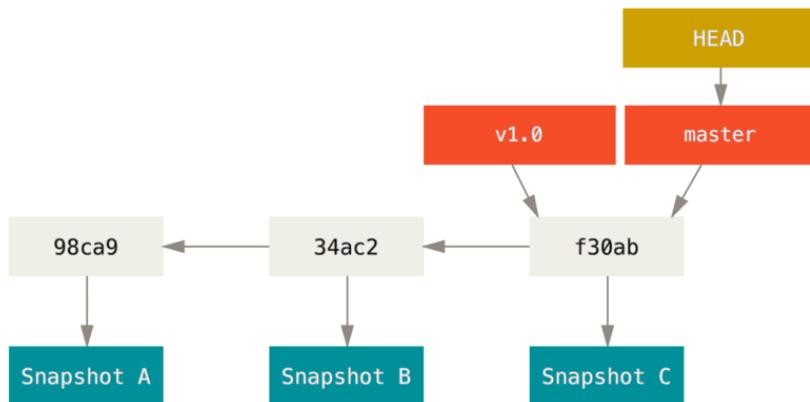


Рис. 18.2 Гілка та її історія фіксацій

18.2. Робота з гілками

Створення нової гілки виконується командою:

```
$ git branch testing
```

Наприклад нова гілка під назвою **testing** (рис. 18.3)

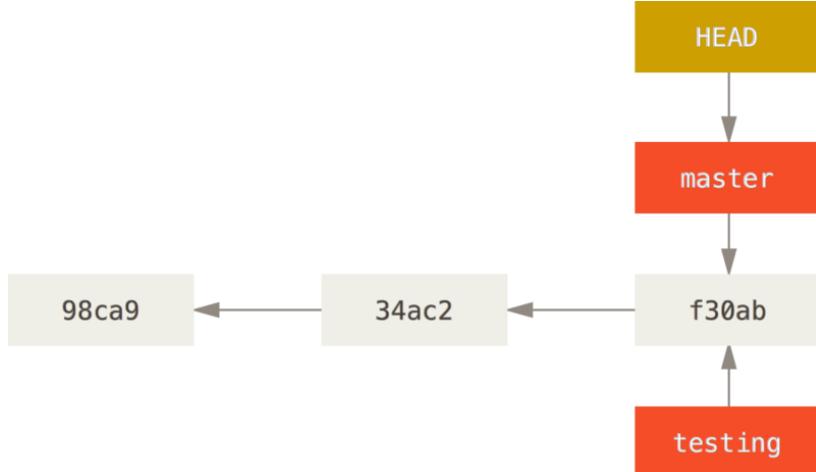


Рис. 18.3 Дві гілки вказують на одну послідовність фіксацій. HEAD вказує на поточну гілку

Перехід на існуючу гілку, виконують командою **git checkout** (рис. 18.4).

```
$ git checkout testing
```



Рис. 18.4 HEAD вказує на поточну гілку.

При фіксації поточна гілка пересувається вперед (**testing**) (рис. 18.5).

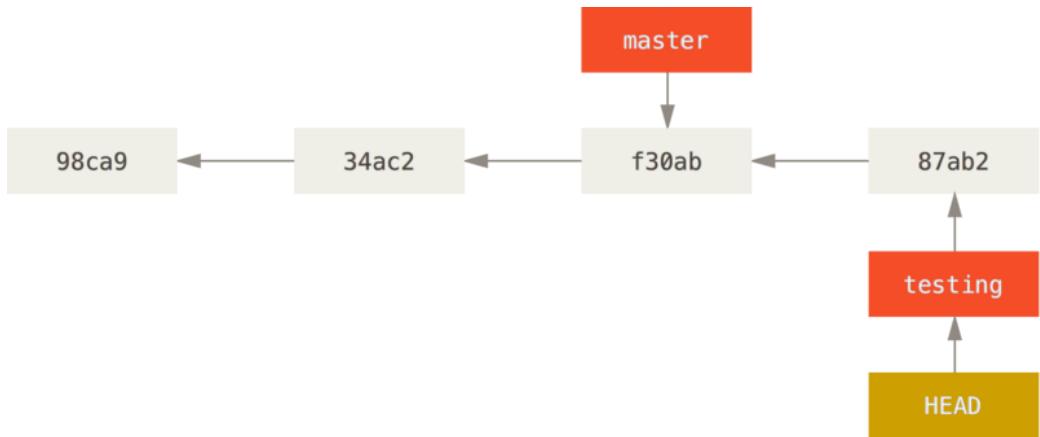


Рис. 18.5 Гілка HEAD пересувається уперед при фіксації

Переключення до гілки **master** виконують командою **git checkout** (рис. 18.6):
`$ git checkout master`

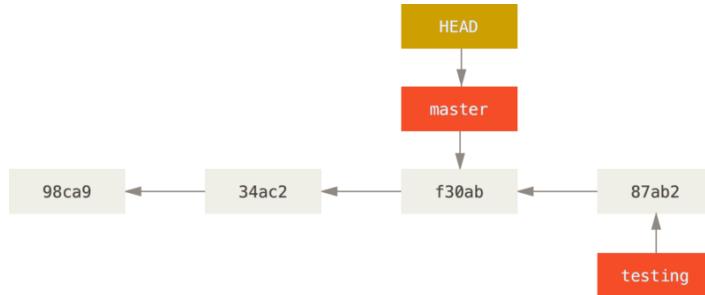


Рис. 18.6 HEAD пересувається, коли ви отримуєте (checkout)

Якщо знову зафіксувати то отримуємо розбіг (**diverged**) (рис 18.7) історії проєкту.

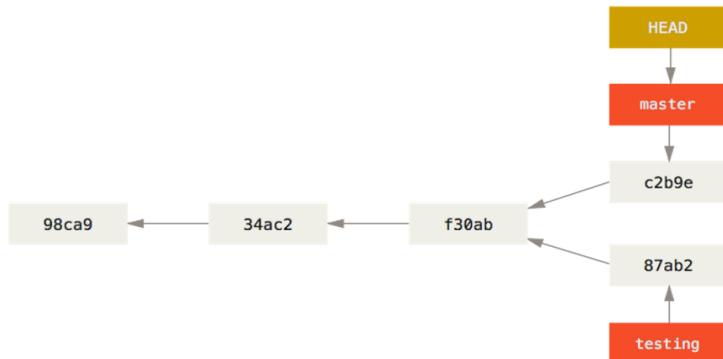


Рис. 18.7 Розбіг історії проєкту

Перевірити історію ваших **commit** ви, можете командою **git log**.

Перевага **Git** над іншими СКВ у простій та зручній роботі з гілками. Гілка в **Git** — це насправді простий файл, що містить 50 символів контрольної суми **SHA-1 commit**, на який вказує. Гілки легко створювати та знищувати. Створити гілку так же швидко, як записати 41 байт до файлу (40 символів та символ нового рядка).

Ми маємо певну розгалужену структуру проекту, наприклад зображену на рис. 18.8.

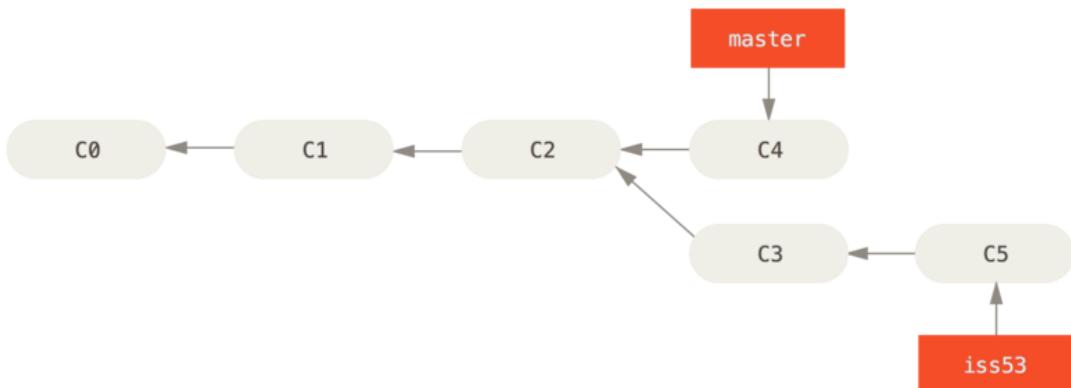


Рис. 18.8 Структура проекту з розгалуженням на гілку `iss53`

Якщо в процесі роботи над проектом – над гілкою `iss53`, ми вирішили що роботу завершено і можна злити з гілкою `master`. Для цього потрібно перейти на робочу гілку та виконати команду **git merge**:

```
$ git checkout master  
Switched to branch 'master'  
$ git merge iss53  
Merge made by the 'recursive' strategy.  
index.html / 1 +  
1 file changed, 1 insertion(+)
```

В цьому випадку **Git** робить просте три-точкове злиття (рис. 18.9), користуючись двома знімками, що вказують на гілки та третім знімком - їх спільним предком.

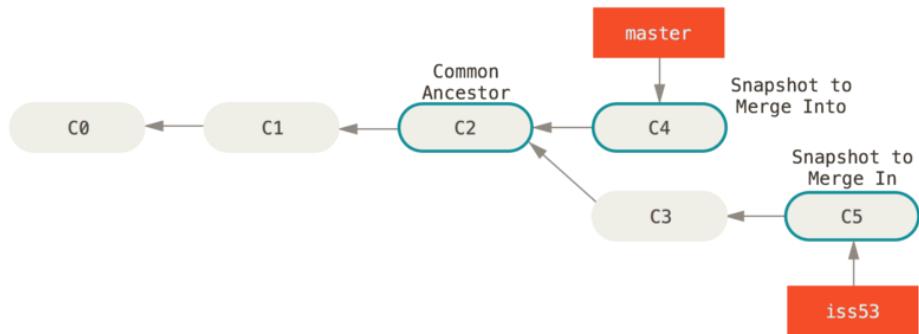


Рис. 18.9 Три відбитки типового злиття

Історія змін двох гілок почала відрізнятися в якийсь момент. Так як **commit** поточної гілки не є прямим нащадком гілки, в яку ви зливаєте зміни, **Git** мусить трохи попрацювати. В цьому випадку **Git** робить просте три-точкове злиття, користуючись двома знімками, що вказують на гілки та третім знімком - їх спільним предком. Такий **commit** називають **commit злиття (merge commit)** (рис. 18.9). Його особливістю є те, що він має більше одного батьківського **commit**.

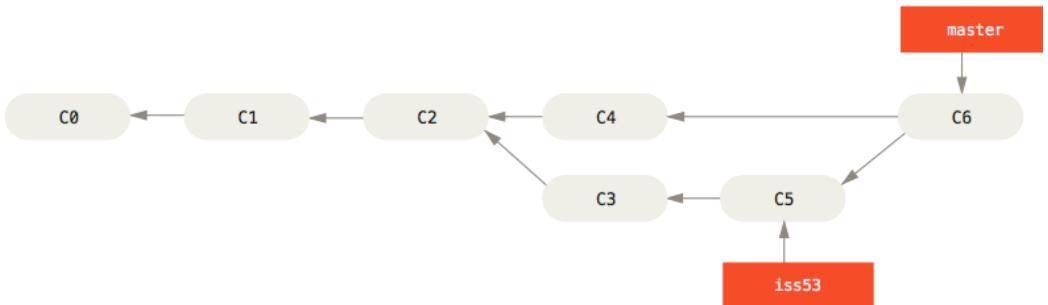


Рис. 18.9 Commit злиття

Тепер, коли ваші зміни зафіксовані, гілка **iss53** вам більше не потрібна. Її можна видалити:

```
$ git branch -d iss53
```

Git сам визначає найбільш підходящого спільногопадкоємця, якого брати за основу зливання. Це відрізняє **Git** від старіших систем таких як **CVS** чи **Subversion** (до версії 1.5), де розробник, що виконує зливання, сам повинен вказувати що брати за основу зливання.

18.3. Кроки реалізації іт-проектів в GitHub

З розвитком систем контролю версій з'явилися різні стилі розробки, що дозволяють програмістам простіше знаходити баги, писати код паралельно з колегами і підвищувати частоту релізів. Сьогодні більшість програмістів використовують для створення якісного програмного забезпечення одну із двох моделей розробки: **Git-flow** або **магістральну розробку**.

Робочий процес Git-flow, який був популяризований першим, є суворішою моделлю розробки, в якій схвалювати зміни основного коду можуть лише певні люди. Це дозволяє забезпечити якість коду та звести кількість багів до мінімуму.

Магістральна розробка – більш відкрита модель, оскільки доступ до основного коду мають усі розробники. Це дозволяє командам швидко виконувати ітерації та впроваджувати CI/CD.

У розробці ПЗ, **CI/CD** або **CICD** – це комбінація безперервної інтеграції (*continuous integration*) та безперервного розгортання (*continuous delivery* або *continuous deployment*) програмного забезпечення у процесі розробки.

CI/CD поєднує розробку, тестування та розгортання додатків.

На даний момент DevOps-програмісти прагнуть застосовувати CI/CD практично для всіх завдань

Магістральна розробка - це метод контролю версій, при якому розробники поєднують невеликі часті оновлення з центральною магістраллю або основною гілкою. Це звичайна практика серед команд DevOps та один із етапів життєвого циклу DevOps, оскільки цей метод дозволяє оптимізувати етапи злиття та інтеграції. Магістральна розробка є обов'язковою практикою CI/CD. В даному випадку розробники можуть створювати короткострокові гілки з декількома невеликими комітами, на відміну від інших стратегій із функціональними гілками, які надовго залишаються у роботі. У міру зростання складності бази коду та розміру команди модель магістральної розробки допомагає підтримувати надходження релізів до робочого середовища.

Git-flow – альтернативна модель розгалуження Git, в якій використовуються довгострокові функціональні гілки та декілька основних гілок. У Git-flow використовується більше гілок, їх термін життя довше, а комміти в них більші, ніж у моделі магістральної розробки. Відповідно до цієї моделі розробники створюють функціональну гілку і відкладають її злиття з головною магістральною гілкою до завершення роботи над функцією. Такі довгострокові функціональні гілки вимагають

тіснішої взаємодії розробників при злитті, оскільки створюють підвищений ризик відхилення від магістральної гілки та виконання конфліктуючих оновлень.

Модель Git-flow також має на увазі окремі напрямки основних гілок для розробки, термінових виправлень, функцій та релізів. Для злиття коммітів між цими гілками застосовуються різні стратегії. Оскільки більша кількість гілок створює складності при взаємодії та управлінні, у таких системах часто виникає потреба у додаткових нарадах щодо планування та перевірок з боку команди.

Вести магістральну розробку набагато простіше, оскільки як джерело виправлень і релізів виступає основна гілка. Даня модель розробки передбачає, що основна гілка завжди стабільна, не має помилок і готова до розгортання.

Магістральна технологія потрібна для безперервної інтеграції. Якщо процеси складання та тестування автоматизовані, але розробники довго працюють із ізольованими функціональними гілками, які рідко інтегруються у загальну гілку, потенціал безперервної інтеграції залишається нереалізованим.

Магістральна технологія полегшує інтеграцію коду. Коли розробники завершують новий етап роботи, вони повинні виконати злиття нового коду з головною гілкою. Однак злиття змін з магістральною гілкою не виконується доти, доки не буде підтверджено їх успішне складання. На цьому етапі можуть виникати конфлікти, якщо з початку нової роботи в магістральну гілку були внесені зміни. У міру розширення команд розробників та масштабування бази коду ці конфлікти стають дедалі складнішими. Таке трапляється, коли розробники створюють окремі гілки з відхиленнями від вихідної гілки, а інші розробники одночасно виконують злиття коду, що перекривається. На щастя, магістральна технологія скорочує такі конфлікти.

Основні переваги магістрального підходу:

- Неперервна інтеграція коду.
- Неперервна перевірка коду.
- Послідовні релізи коду в робоче середовище.
- Магістральна розробка та CI/CD

Модель магістральної розробки передбачає наявність репозиторію зі стабільним потоком комітів, які надходять у основну гілку. Безперервна інтеграція досягається шляхом додавання набору автоматизованих тестів та моніторингу покриття коду для цього потоку комітів. Після злиття нового коду із магістраллю виконується перевірка його якості. Для цього запускаються автоматизовані тести з інтеграції та покриття коду.

При частих та невеликих комітах у системах магістральної розробки перевірка коду стає більш ефективною. При малому розмірі гілок розробники можуть швидко

переглядати та перевіряти невеликі зміни. Це набагато простіше, ніж працювати з довгостроковою функціональною гілкою, в якій перевіряльнику доводиться читати кілька сторінок коду або вручну перевіряти велику область змін коду.

Командам слід виконувати часті щоденні злиття з основною гілкою. Ця модель розробки передбачає, що магістральна гілка залишається зеленою, тобто готова до розгортання на будь-якому коміті. Автоматизовані тести, злиття коду та його перевірка дозволяють гарантувати, що код проєкту, що розробляється за моделлю з магістральною гілкою, готовий до розгортання у робочому середовищі у будь-який час. Завдяки цьому команда отримує можливість виконувати часті розгортання у робочому середовищі та ставити подальші цілі щоденних релізів.

Магістральна розробка та CI/CD

У міру зростання популярності безперервної інтеграції та безперервного постачання моделі розгалуження були вдосконалені та оптимізовані, що призвело до поширення розробки на основі магістральної гілки. Сьогодні магістральна розробка є обов'язковою умовою безперервної інтеграції. У системах безперервної інтеграції розробники застосовують магістральну розробку разом із автоматичними тестами, виконуваними після кожного коміті у магістральну гілку. Це гарантує, що проєкт залишається працездатним будь-якої миті.

Магістральна розробка дозволяє командам швидко та послідовно випускати код.

Прийоми та методи, які допоможуть скоротити робочий цикл команди та оптимізувати графік релізів:

- Розробка невеликими пакетами.
- Прапорці функцій.
- Впровадження комплексного автоматизованого тестування.
- Впровадження асинхронної перевірки коду.
- Впровадження в репозиторії коду програми не більше трьох активних гілок.
- Впровадження злиття в магістральну гілку не рідше одного разу на день.
- Скорочення кількості заборон на зміну коду та етапи інтеграції.
- Виконання складання швидко та тестування негайно.

Магістральна розробка підтримує швидкий цикл постачання коду до робочого середовища. Якщо порівнювати таку модель розробки з музикою, вийде швидке стакато: короткі, лаконічні ноти, які швидко змінюють одна одну, і ці ноти — коміті в репозиторій. Малий розмір комітів та гілок забезпечує прискорений темп злиття та розгортання.

Невеликі зміни в результаті кількох комітів або модифікація декількох рядків коду зводять до мінімуму когнітивні витрати. Командам набагато простіше вести змістовні обговорення та швидко приймати рішення при розгляді обмеженої області коду, а не великої кількості змін.

Прапорці можливостей чудово доповнюють модель магістральної розробки, дозволяючи розробникам поміщати нові зміни до неактивного шляху коду та активувати його пізніше. Це дає можливість відмовитися від створення функціональної гілки в окремому репозиторії і натомість робити комміти для нової функції безпосередньо в основну гілку, поміщаючи їх у шлях, позначений прапорцем можливості.

Функціональні прапори підтримують принцип оновлення невеликими пакетами. Замість створення функціональної гілки та очікування складання повної специфікації, розробники можуть виконати коміт у магістральну гілку, вводячи прапор функції, та потім будуть відправляти нові комміти магістральної гілки, у яких специфікація функції реалізується у межах прапора.

Автоматичне тестування необхідне для будь-якого сучасного проєкту розробки з методології CI/CD. Існують різні типи автоматичних тестів, що виконуються на різних етапах конвеєра релізів. Короткострокові модульні та інтеграційні тести виконуються у процесі розробки та після злиття коду. Більш тривалі та комплексні наскрізні тести виконуються на пізніших етапах конвеєра на основі всього розділу проіндексованих файлів або робочого середовища.

Автоматичні тести запускаються для невеликих пакетів зі злиття розробниками нових коммітів, підтримуючи процес магістральної розробки. Набір автоматичних тестів перевіряє код на наявність будь-яких проблем та автоматично підтверджує чи відхиляє його. Це дозволяє розробникам швидко створювати комміти та автоматично тестувати їх, щоб дізнатися, чи не виникли нові проблеми.

При магістральній розробці перевірка коду має виконуватися негайно, а не плануватися на майбутнє в асинхронної системі. Автоматичні тести утворюють рівень запобіжної перевірки коду. Коли розробники будуть готові перевірити запит **pull** від учасника команди, вони зможуть спочатку подивитися, чи позитивний результат автоматичних тестів та чи розширилося покриття коду. Так перевірлюнки можуть переконатися, що новий код відповідає певним специфікаціям. Після цього перевірку можна звести до оптимізації.

Після злиття гілки її рекомендується видалити. Репозиторій з великою кількістю активних гілок має низку неприємних побічних ефектів. Командам може бути корисно дізнатися з активних гілок, яка робота ведеться, проте за наявності

застарілих та неактивних гілок ця можливість зникає. Деякі розробники використовують інтерфейси Git, які при завантаженні великої кількості віддалених гілок починають працювати повільно.

Високопродуктивні команди, що застосовують модель магістральної розробки, повинні закривати та об'єднувати будь-які відкриті та готові до злиття гілки хоча б раз на день. Це допомагає підтримувати періодичність та задає графік відстеження релізів. Потім наприкінці дня команда може відзначити магістральну гілку як коміт релізу. Це має корисний побічний ефект у вигляді генерації щоденного гнучкого інкременту релізу.

У командах, що наслідують принципи AGILE та практикують безперервну інтеграцію та безперервне постачання, не повинно виникати потреби у планових заборонах на зміни коду або у перервах на етапі інтеграції, хоча організація може використовувати ці механізми з інших причин. У CI/CD принцип безперервності передбачає, що оновлення виконуються постійно. Команди, які ведуть магістральну розробку, повинні по можливості уникати заборон на зміни коду, які зупиняють усі процеси, та планувати свою роботу так, щоб конвеєр релізів не зупинявся.

Для підтримки високої частоти релізів необхідно оптимізувати час складання та тестування. Інструменти збирання в CI/CD повинні в міру необхідності застосовувати шари кешування, щоб уникнути дорогих обчислень на основі статичних ресурсів. Тести мають бути оптимізовані так, щоб не виконувати зайвих звернень до сторонніх служб.

Магістральна розробка сьогодні є стандартом для високопродуктивних команд розробки, оскільки такий підхід задає та підтримує високу частоту релізів на основі спрощеної стратегії розгалуження Git. Крім того, магістральна розробка забезпечує командам розробки більшу гнучкість та контроль над процесом постачання програмного забезпечення кінцевому користувачеві.

18.3. Робочий процес Git-flow Workflow

Git-flow - це застаріла версія робочого процесу Git, яка свого часу стала принципово новою стратегією управління гілками в Git. Популярність Git-flow стала знижуватися під впливом магістральних робочих процесів, які сьогодні вважаються кращими для сучасних схем безперервної розробки ПЗ і застосування DevOps. Крім того, Git-flow не надто зручно застосовувати у процесах CI/CD.

Git-flow – альтернативна модель розгалуження **Git**, в якій використовуються функціональні гілки та декілька основних гілок. Ця модель була вперше опублікована

та популяризована Вінсентом Дрісеном на сайті [nvie](#). У порівнянні з моделлю магістральної розробки, в **Git-flow** використовується більше гілок, кожна з яких існує довше, а коміти зазвичай більші. Відповідно до цієї моделі розробники створюють функціональну гілку і відкладають її злиття з головною магістральною гілкою до завершення роботи над функцією. Такі довгострокові функціональні гілки вимагають тісної взаємодії розробників при злитті та створюють підвищений ризик відхилення від магістральної гілки. В них також можуть бути конфліктні оновлення.

Git-flow можна використовувати для проектів, у яких заплановано цикл релізів та реалізується характерна для **DevOps** методика безперервного постачання. У цьому процесі використовуються поняття та команди, які були запропоновані в рамках робочого процесу з функціональними гілками. Однак **Git-flow** привносить нові специфічні ролі для різних гілок та визначає характер та частоту взаємодії між ними. Крім функціональних гілок у рамках цього робочого процесу використовуються окремі гілки для підготовки, підтримки та реєстрації релізів. При цьому ви, як і раніше, можете користуватися перевагами робочого процесу з функціональними гілками, такими як запити pull, ізольовані експерименти та ефективна командна взаємодія.

Git-flow – це методика роботи з **Git**; що визначає, які види гілок необхідні проекту та як виконувати злиття між ними. Набір інструментів git-flow є окремою утилітою командного рядка, яка вимагає установки.

Установку у системах OS X можна виконати командою `brew install git-flow`. Якщо ви використовуєте Windows, вам потрібно завантажити та встановити **git-flow**. Після встановлення рішення **git-flow** необхідно виконати команду **git-flow init**, щоб використовувати його у проекті. Цей набір інструментів відіграє роль обгортки Git. Команда **git-flow init** є розширенням стандартної команди **git init** і не вносить змін до репозиторій, крім створення гілок.

Гілка розробки та головна гілка в Git-flow

Git-flow реєструє історію проекту починаючи з двох гілок – головної (**main**), що зберігає офіційну історію релізу та гілки розробки (**develop**). Для зручності рекомендується надавати всім коммітам у гілці **main** номер версії.

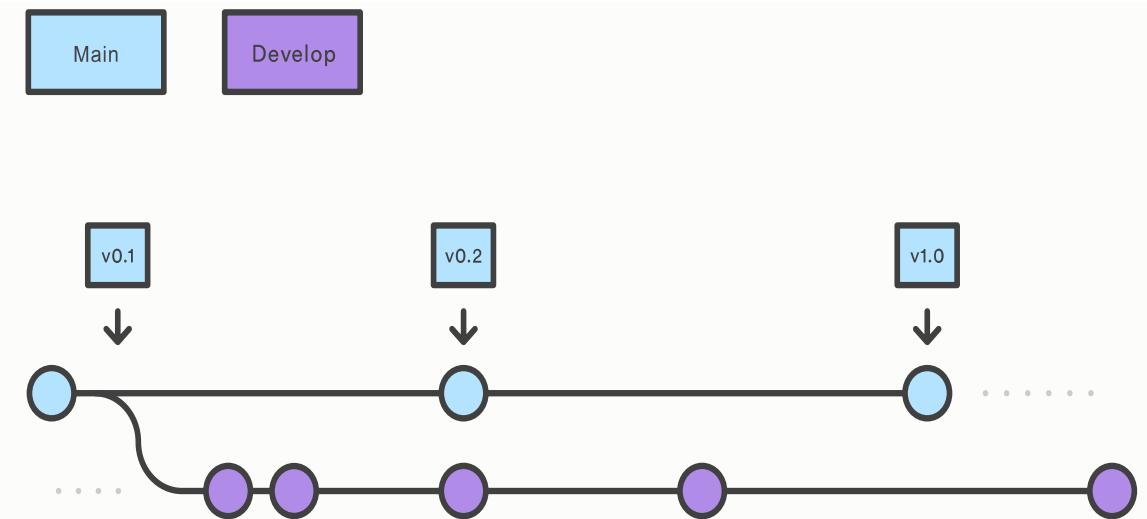


Рис. 18.10 Дві головні гілки початку проекту

Перший крок робочого процесу полягає у створенні гілки **develop** від стандартної гілки **main** (рис. 18.10). Розробнику буде простіше створити порожню гілку **develop** локально та відправити її на сервер.

```
git branch develop
git push -u origin develop
```

Гілка **develop** буде зберігати повну версію проекту, а **main** скорочену.

При використанні бібліотеки розширення **git-flow**, для створення гілки **development** можна виконати команду **git flow init** в існуючих репозиторіях: При використанні бібліотеки розширення **git-flow**, для створення гілки **development** можна виконати команду **git flow init** в існуючих репозиторіях.

Функційні гілки (feature)

Під кожну нову функцію виділяють власну гілку, яку можна відправити в центральний репозиторій для створення резервної копії команди або спільної роботи. Гілки функцій створюються не на основі **main**, а на основі **development**. Коли робота над функцією завершується, відповідна вітка зливається з гілкою **development**. Функції не слід відправляти безпосередньо у гілку **main**.

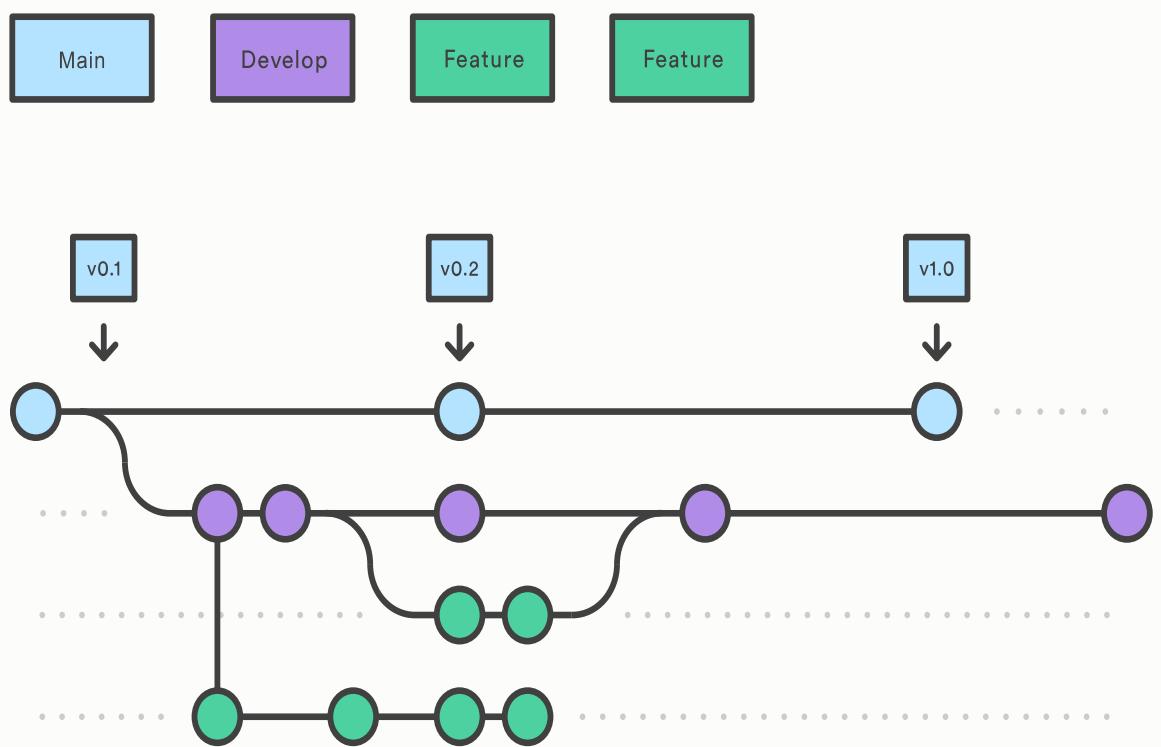


Рис. 18.11 Функційні гілки

Зверніть увагу, що комбінація гілок **feature** з гілкою **develop** фактично є робочим процесом з функціональними гілками (рис. 18.11). Але робочий процес **Git-flow** на цьому не закінчується. Як правило, гілки **feature** створюються на основі останньої гілки **develop**.

Створення функційної гілки

Без застосування розширення **Git-flow**.

```
git checkout develop
git checkout -b feature_branch
```

З застосуванням розширення **Git-flow**.

```
git flow feature start feature_branch
```

Після роботи над функцією слід об'єднати гілку **feature_branch** з **develop**.

Без застосування розширення **Git-flow**.

```
git checkout develop  
git merge feature_branch
```

З застосуванням розширення Git-flow.

```
git flow feature finish feature_branch
```

Гілки випуску (release)

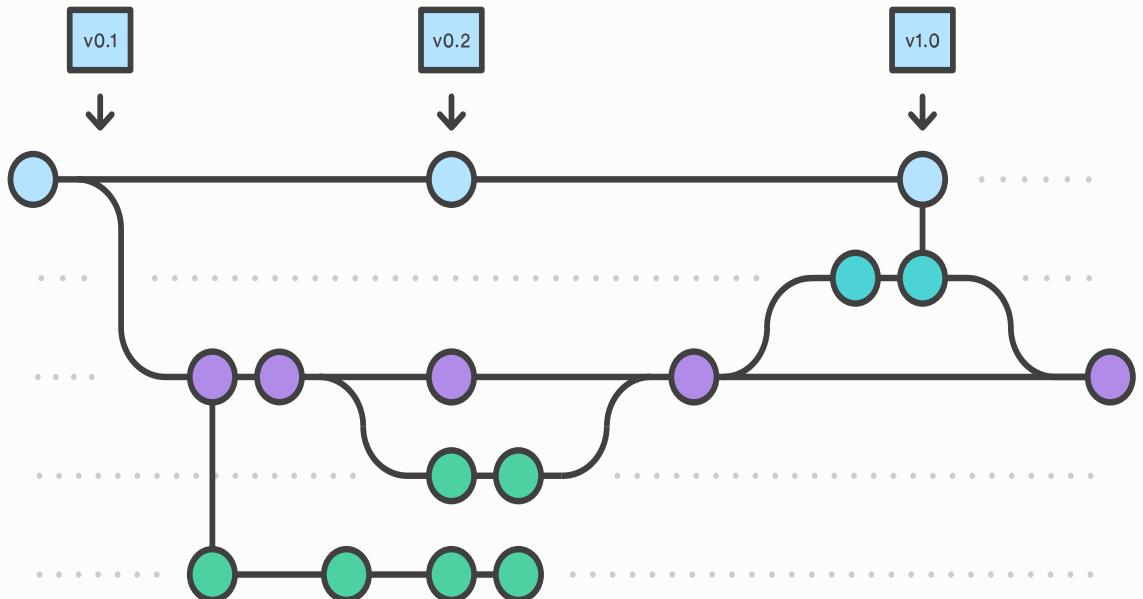


Рис. 18.12 Гілки випуску

Коли у гілці **develop** виявляється достатньо функцій для випуску (або наближається призначена дата релізу), від гілки **develop** створюється гілка **release** (рис. 18.12). Створення цієї галузі запускає наступний цикл релізу, і з цього моменту нові функції додати більше не можна — допускається лише виправлення багів, створення документації та вирішення інших завдань, пов'язаних з релізом. Коли підготовка до постачання завершується, гілка **release** зливається з **main** і присвоюється

номер версії. Крім того, потрібно виконати її злиття з гілкою **develop**, в якій з моменту створення гілки релізу могли виникнути зміни.

Завдяки тому, що для підготовки випусків використовується спеціальна гілка, одна команда може доопрацьовувати поточний випуск, тоді як інша команда продовжує роботу над функціями наступного. Це також дозволяє розмежувати етапи розробки (наприклад, можна легко присвятити тиждень підготовці до версії 4.0 і дійсно побачити це в структурі репозиторію).

Створення гілок **release** це ще одна проста операція розгалуження. Як і гілки **feature**, гілки **release** засновані на гілці **develop**. Створити нову гілку **release** можна за допомогою наступних команд.

Без застосування розширення **Git-flow**.

```
git checkout develop  
git checkout -b release/0.1.0
```

З застосуванням розширення **Git-flow**.

```
$ git flow release start 0.1.0  
Switched to a new branch 'release/0.1.0'
```

Коли підготовка до постачання завершується, реліз зливається з гілками **main** і **develop**, а гілка **release** видаляється. Важливо злити її з гілкою **develop**, оскільки у гілку **release** могли додати критичні оновлення, які мають бути доступні для нових функцій. Якщо у вашій організації приділяють особливу увагу перевірці коду, це ідеальне місце для виконання запиту **pull**.

Для завершення роботи у гілці **release** використовуйте такі команди:

Без застосування розширення **Git-flow**.

```
git checkout main  
git merge release/0.1.0
```

З застосуванням розширення **Git-flow**.

```
git flow release finish '0.1.0'
```

Гілки виправлення (hotfix)

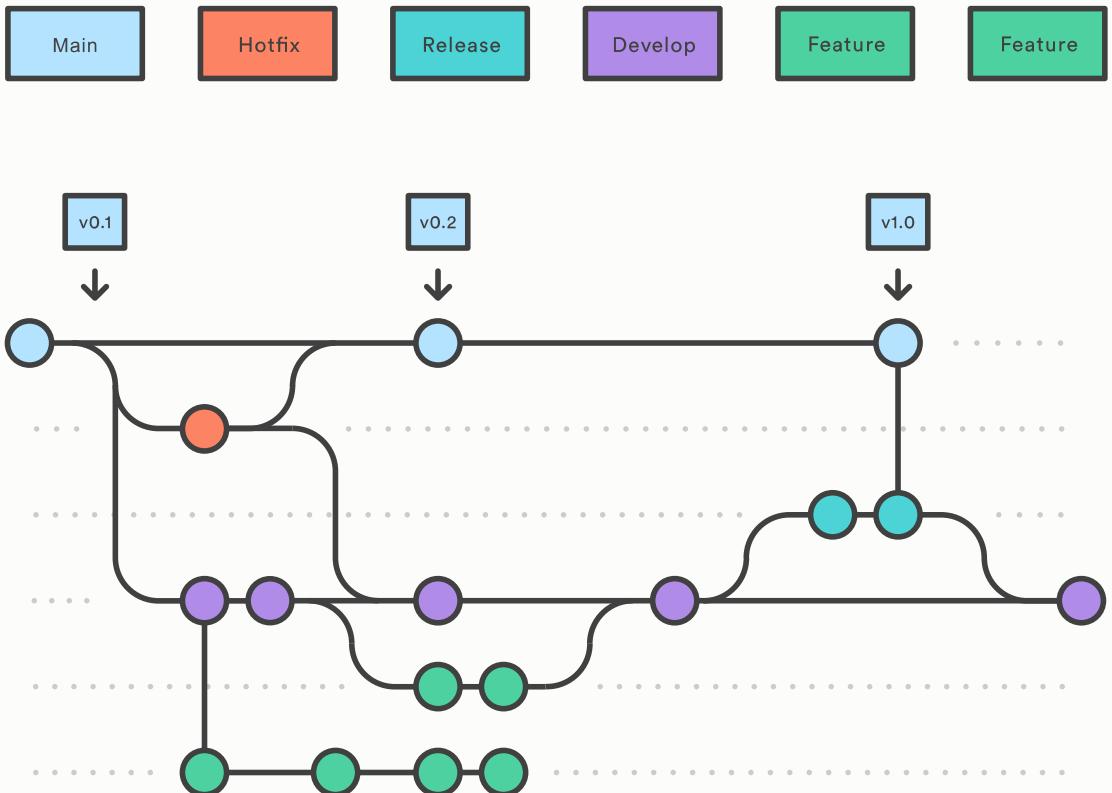


Рис. 18.13 Гілки виправлення

Гілки супроводу або виправлення (**hotfix**) використовуються для швидкого внесення виправлень у робочі релізи (рис. 18.3). Гілки **hotfix** дуже схожі на гілки **release** і **feature**. Відмінність полягає в тому, що вони створюються на основі **main**, а не **develop**. Це єдина гілка, яку потрібно обов'язково створювати безпосередньо від **main**. Як тільки виправлення завершено, цю гілку слід злити з **main** і **develop** (або поточного гілкою **release**), а гілці **main** присвоїти оновлений номер версії.

Завдяки спеціальній гілці для виправлення помилок команда може усувати проблеми, не перериваючи рушту робочого процесу і не чекаючи наступного циклу релізу. Гілки супроводу можна розглядати як спеціальні гілки **release**, які працюють безпосередньо з **main**. Гілку **hotfix** можна створити за допомогою наступних команд.

Без застосування розширення **Git-flow**.

```
git checkout main  
git checkout -b hotfix_branch
```

З застосуванням розширення **Git-flow**.

```
$ git flow hotfix start hotfix_branch
```

Після завершення роботи з гілкою **hotfix** її зливають з **main** і **develop** (як і у випадку з гілкою **release**).

```
git checkout main
git merge hotfix_branch
git checkout develop
git merge hotfix_branch
git branch -D hotfix_branch
```

```
$ git flow hotfix finish hotfix_branch
```

Приклад

Розглянемо повний цикл роботи з функціональною гілкою (передбачається, що ми маємо репозиторій з гілкою **main**).

```
git checkout main
git checkout -b develop
git checkout -b feature_branch
# work happens on feature branch
git checkout develop
git merge feature_branch
git checkout main
git merge develop
git branch -d feature_branch
```

Крім роботи з гілками **feature** та **release**, продемонструємо роботу з гілкою **hotfix**:

```
git checkout main
git checkout -b hotfix_branch
# work is done commits are added to the hotfix_branch
git checkout develop
git merge hotfix_branch
git checkout main
git merge hotfix_branch
```

Ключові ідеї, які потрібно запам'ятати про **Git-flow**:

- Ця модель відмінно підходить для організації робочого процесу на основі релізів.
- Робота за моделлю **Git-flow** передбачає створення спеціальної гілки для виправлення помилок у робочому релізі.

Послідовність дій під час роботи за моделлю Git-flow:

1. З гілки **main** створюється гілка **develop**.
2. З гілки **develop** створюється гілка **release**.
3. З гілки **develop** створюються гілки **feature**.
4. Коли робота над гілкою **feature** завершується, вона зливається у гілку **develop**.
5. Коли робота над гілкою **release** завершується, вона зливається з гілками **develop** та **main**.
6. Якщо у гілці **main** виявляється проблема, з **main** створюється гілка **hotfix**.
7. Коли робота над гілкою **hotfix** завершується, вона зливається з гілками **develop** та **main**.

Питання до розділу 18

1. Зміст об'єкту фіксації Git?
2. Опишіть фіксацію об'єктів в Git/
3. Створення нової гілки в Git?
4. Переключення гілок в Git?
5. Зливання гілок в Git?
6. Які підходи до ведення в системі керування версіями проекту ви знаєте?
7. Що представляє магістральний підхід?
8. Що представляє **Git-flow** підхід?
9. В чому переваги магістрального підходу?

10. Які ви знаєте рекомендації до магістрального підходу в веденні проекту?
11. Які гілки ведення проекту в **Git-flow** ви знаєте?
12. В чому головні особливості для функційних, релізних та гілках виправлення в проекті?
13. Перелічіть послідовність дій під час роботи з **Git-flow**?

ПЕРЕЛІК ПОСИЛАНЬ

1. Кузьміних В.О., Тараненко Р.А. Основи управління ІТ проектами // Навч. посіб. для студ. 122 «Комп’ютерні науки» / КПІ ім. Ігоря Сікорського. Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 9 від 29.05.2019 р.) – Київ : КПІ ім. Ігоря Сікорського, 2019. – 76 с.
2. Eric S. Norman. Work Breakdown Structure. ISBN 9780470177129; 2008 г. – 304 с.
3. Кузьміних В.О., Коваль О.В., Воронько М.П. Оцінка часу виконання типових задач проектів на підприємствах з функціональною організаційною структурою// Реєстрація, зберігання і обробка даних,ISSN 1560-9189, 2012 т. 14, № 3, с.77-82.
4. Кузьміних В.О., Хаустов Д.В., Коростельова Є.Ю. Аналіз ризиків у корпоративної системі управління проектами// Реєстрація, зберігання і обробка даних, 2010 . – т. 12, № 3, с. 99–107.
5. Кузьміних В.О., Коваль О.В. Реалізація сценарного підходу в управлінні проектами на основі типових задач // Реєстрація, зберігання і обробка даних, ISSN 1560-9189, 2015.т. 17, №1,с.77-87.
6. Кузьміних В.О. Трирівнева корпоративна система управління проектами//Реєстрація, зберігання і обробка даних, 2009, т.11, №3, с. 75–82.
7. Кузьміних В.О., Отрох С.І., Воронько М.П., Тараненко Р.А. Fundamentals of IT project management // Навч. посіб. для студ. 122 «Комп’ютерні науки» / КПІ ім. Ігоря Сікорського. Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 7 від 27.02.2020 р.) – Київ : КПІ ім. Ігоря Сікорського, 2019. – 74 с.
8. Ярошенко Ф. А., Бушуев С. Д., Танака Х. — Управление инновационными проектами и программами на основе системы знаний Р2М. - К.: 2011. 268 с.
9. Керівництво з управління інноваційними проектами і програмами організацій: Монографія / Переклад на українську мову під ред. проф. Ярошенка Ф.О. – К.: Новий друк. – 2010. – 160 с.
10. Ноздріна Л.В. Управління проектами: підручник / Ноздріна Л.В., Ящук В.І., Полотай О.І./ За заг.ред. Л.В. Ноздріної. - К.: Центр учебової літератури, 2010. – 432 с.
11. Крайнік О.М.. Планування проектних дій: навчально-методичний посібник для студентів ЗДІА спеціальності 8.18010013“Управління проектами» dennої форми навчання / О.М. Крайнік, Н.І. Тахтаджиєва - Запоріжжя, ЗДІА, 2015. - 80 с.

12. «Управління проектами»: навчальний посібник / Уклад.: Л.Є. Довгань, Г.А.Мохонько, І.П Малик. – К.: КПІ ім. Ігоря Сікорського, 2017. – 420 с.
13. Стандарт з управління проектами та Настанова до зводу знань з управління проектами (PMBOK Guide 7) / Project Management Institute, 2022–274 с.
14. Блага Н. В. Управління проектами : навчальний посібник / Наталія Блага. - Львів: Львівський державний університет внутрішніх справ, 2021. - 152 с.
15. Skott Chacon, Ben Straub. Pro Git. 2nd Edition : Apress, 2014. URL: <https://git-scm.com/book/uk/v2> (Дата звернення 12.12.2022).
16. Директория Git и рабочая директория. URL: <https://cutt.us/ah2VE> (Дата звернення 12.12.2022).
17. Объектная модель Git. URL: <https://cutt.us/Sg23V> (Дата звернення 12.12.2022).
18. Roles in an organization. URL: <https://docs.github.com/en/organizations/managing-peoples-access-to-your-organization-with-roles/roles-in-an-organization> (Дата звернення 12.12.2022).
19. Repository roles for an organization. URL: <https://docs.github.com/en/organizations/managing-access-to-your-organizations-repositories/repository-roles-for-an-organization> (Дата звернення 12.12.2022).

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. ДСТУ ISO 9000:2007. Системи управління якістю. Основні положення та словник термінів.
2. Burke R. Project management: planning and control techniques. New Jersey, USA, 2013.
3. Heagney J. Fundamentals of project management. Amacom, 2016.
4. Данченко О. Б., Занора В. О. Проектний менеджмент: управління ризиками та змінами в процесах прийняття управлінських рішень : монографія / О. Б. Данченко, В. О. Занора. – Черкаси : ПП Чабаненко Ю.А., 2019. – 278 с
5. Єгорченкова Н.Ю., Катаєва Є.Ю. Азбука управління проектами. Планування: навч. посіб. К.: КНУ ім. Т.Шевченка, 2017. 117 с
6. Петрович Й.М., Новаківський І.І. Управління проектами: підруч. Нац. ун-т «Львів. політехніка». Львів: Вид-во Львів. політехніки, 2018. 395 с
7. Старченко Г.В. Управління проектами: теорія та практика: навч. посіб. Чернігів. нац. технол. ун-т. Чернігів: Брагинець О.В. [вид.], 2018. 304 с.
8. Яковенко О. І. Управління проектами та ризиками: навч. посіб. Ніжин: Лисенко М.М., 2019. 196 с.
9. Управління інноваційними проектами в умовах міжнародної інтеграції: моногр. / О.О. Охріменко та ін. Нац. техн. ун-т України «Київ. політехн. ін-т ім. Ігоря Сікорського». Київ: КПІ ім. Ігоря Сікорського, 2018. 260 с.
10. Галушка З.І, Волощук О.А. Управління проектами. Project management: навч. посіб. Чернів. нац. ун-т ім. Юрія Федьковича. Чернівці: ЧНУ ім. Ю. Федьковича: Рута, 2018. 119 с.
11. Єгорченкова Н.Ю., Катаєва Є.Ю. Азбука управління проектами. Планування: навч. посіб. К.: КНУ ім. Т.Шевченка, 2017. 117 с
12. Гордієнко В.О. Управління інноваційними проектами і програмами: навч. посіб. Ун-т мит. справи та фінансів. Дніпро: Ун-т мит. справи та фінансів, 2019. 115 с.
13. Управління змінами та проектами: навч. посіб. / Грибик І. І. та ін. Нац. ун-т «Львів. політехніка». Львів: Центр Європи, 2017. 168 с.
14. Приймак В.М. Управління проектами: навч. посіб. Київ: КНУ ім. Тараса Шевченка, 2017. 459 с.
15. Петренко Н.О., Кустріч Л.О., Гоменюк М.О. Управління проектами: навч. посіб. К.: «Центр учебової літератури», 2015. 244 с.

16. Кузьмичов А.І. Планування та управління проектами. Моделювання засобами MS Excel: практикум. Ін-т проблем реєстрації інформації НАН України. Київ: Ліра-К, 2016. 179 с.
17. Інструменти для роботи в команді: від стартапу до великої корпорації. URL: <https://www.atlassian.com/ru/software/jira> (Дата звернення 12.05.2023).
18. Українська асоціація управління проектами. URL: <http://www.upma.kiev.ua/> (Дата звернення 12.05.2023).
19. Что такое контроль версий? URL: <https://cutt.us/AeHN8> (Дата звернення 12.12.2022).
20. Вступ – про систему контроля версій. URL: <https://cutt.us/LERkK> (Дата звернення 12.12.2022).
21. Якоб Стопак. История систем контроля версий. URL: <https://cutt.us/76mPT> (Дата звернення 12.12.2022).
22. Налаштування Git – Конфігурація Git. URL: <https://cutt.us/GomEX> (Дата звернення 12.12.2022).

Додаток 1

Глосарій

ЕРМ (Enterprise Project Management) – сукупність програмних засобів, що забезпечують управління проектами та агрегованими групами проектів (портфелям, програмами, папками) у корпоративній системі і базуються на клієнт-серверній архітектурі.

Вixa проєкту – значна подія в проєкті. Часто позначає зміну фази проєкту. Залежно від типу і виду проєкту можуть бути використані різні структури життєвих циклів проєкту.

Група управління проєктом (далі – ГУП) – тимчасове організаційне формування, яке призначено виконувати функції по управлінню одним конкретним проєктом.

Зацікавлені сторони – Окремі особи й організації, залучені до проєкту, або ті, чиї інтереси можуть позитивно чи негативно вплинути на результати виконання проєкту.

Життєвий шлях (цикл) проєкту (Project Way (Life) Cycle) – Повний набір послідовних фаз проєкту, назва і число яких визначається виходячи з технології виробництва робіт і потреб контролю з боку організації або організацій, залучених в проєкт.

Ініціація проєкту (Project Initiation) – процес управління проєктом, результатом якого є ухвалення рішення про початок проєкту або чергової фази його життєвого циклу.

Керівник групи управління проєктом (ГУП) – Відповідальна особа, яка призначається наказом на створення групи управління проєктом, відповідає за виконання проєкту та виконує роль керівника проєкту.

Керівник проєкту – Відповідальна особа, яка відповідає за виконання проєкту. У разі створення ГУП, ці функції виконує керівник ГУП.

Команда управління проєктом (Project Management Team) – Члени команди проєкту, які безпосередньо залучені в ухвалення рішень за проєктом і процеси управління проєктом.

Координатор проєкту – Особа, відповідальна за оперативне керівництво діями з виконання проєкту (частини проєкту) та підзвітна безпосередньо керівнику проєкту. Роль координатора проєкту може виконувати керівник проєкту

Критерії успіху проекту (Project Success Criteria) - Сукупність показників, які дають можливість судити про успішність виконання проекту.

Критичний шлях проекту (Critical Path) – В мережній моделі (діаграмі) проекту – послідовність робіт і залежностей, що визначає найраніше завершення проекту. Критичний шлях змінюватиметься час від часу залежно від того, чи завершуються роботи досроко або пізніше планових термінів.Хоча звичайно критичний шлях обчислюється для всього проекту, він може бути визначений також для контрольних подій або для під-проектів. Критичний шлях звичайно складають роботи, резерв часу яких мінімальний або рівний нулю.

Локальний проект – Проект, який виконується в межах одного підрозділу та не є обов'язковим для ведення у відповідності до усіх вимог, стандартів та рекомендацій КСУП, але може використовувати, за необхідністю, певні засоби КСУП.

Майстер-план проекту (Master Schedule) – Укрупнений розклад, розклад узагальнюючого рівня, який включає укрупнені роботи (етапи) і ключові події.

Матриця відповідальності (Responsibility Assignment Matrix) – Структура, яка ставить у відповідність організаційній структурі проекту (OBS) структурну декомпозицію робіт (WBS) для призначення відповідальних осіб за роботи (результати) і фази проекту.

Матрична організація (Matrix Organization) – організаційна структура, в якій виконавці можуть бути одночасно задіяні в роботах на декількох проектах і робити звіт перед декількома керівниками. Менеджер проекту розділяє відповідальність з функціональними менеджерами у визначені пріоритетів і керівництві роботами виконавців, залучених в проект. Існують різні типи матриць, відмінні розділенням влади і відповідальності між функціональними керівниками і менеджерами проектів.

Менеджер проекту (Project Manager) – Особа, відповідальна за управління проектом.

Менеджер проекту - це особа, якій замовник або інвестор делегують повноваження по керівництву роботами в рамках проекту: плануванню, контролю і координації робіт всіх учасників проекту

Мережна діаграма проекту. (Project Network Diagram) – Будь-яке схемне представлення логічних взаємозв'язків між роботами проекту. Завжди зображується зліва направо для відображення хронології проекту.

Мережне планування. – Методи планування з використанням техніки мережного моделювання і аналізу, комплексу взаємозв'язаних робіт.

Моніторинг проєкту (Monitoring) – Збір, аналіз даних, представлення звітів по виконанню проєкту, звично порівняно з планом, і, при необхідності, вироблення таких дій, що коректують.

Організаційна структура проєкту (Project organization) – відповідна до проєкту тимчасова організаційна структура, що включає всіх його учасників і створювана для успішного досягнення цілей проєкту (команда проєкту).

Паспорт проєкту (Статут). (Project Charter) – Документ, що описує та затверджує основні частини змісту, термінів та ресурсів проєкту.

Папки проєктів – Агреговані за довільними ознаками групи проєктів з метою їх групового аналізу. Всі проєкти папки проєктів є обов'язковими для ведення в КСУП.

План проєкту (Project Plan) – Формальний, затверджений документ, що використовується для організації і координації виконання і контролю проєкту. Основне положення використування плану проєкту полягає в документуванні планованих припущень і рішень для забезпечення зв'язку між учасниками проєкту і для документування цілей, наочної області, розкладу робіт і вартості проєкту. План проєкту може бути як укрупненим, так і детальним.

Планування якості (Quality Planning) – Визначення стандартів якості, що відносяться до проєкту, і способів відповідності ним.

Планування комунікацій (Communication Planning) – Визначення інформаційних і комунікаційних потреб учасників проєкту і способів їх задоволення.

Планування поставок (контрактів) (Procurement Planning) – Визначення складу необхідних закупівель (контрактів) і їх основних параметрів (терміни, об'єм і т.ін.).

Планування змісту проєкту (Scope Planning) – Розробка документального уявлення і узгодження змісту і меж проєкту, який включає: обґрунтування проєкту, основні результати, цілі і задачі проєкту.

Планування ресурсів (Resource Planning) – Визначає, які ресурси (люди, устаткування, матеріали і ін.), коли і в яких кількостях необхідні для виконання робіт проєкту.

Портфель проєктів (Project Portfolio) - безліч проєктів і програм, з'єднаних для зручностей управління. Примітка 1. Проєкти і програми в портфелі проєктів можуть мати або не мати загальних цілей, але, як правило, мають загальні обмеження по ресурсах.

Предметна область (зміст) проєкту (Project Scope) – Сукупність продуктів і послуг, виробництво яких повинне бути забезпечене в рамках здійснюваного проєкту.

Проєктний офіс – Окремий підрозділ, що виконує завдання з організаційної та методичної підтримки управління проєктами у корпоративній системі управління проєктами (КСУП)

Програма (Program) – Група взаємозв'язаних проєктів і різних заходів, з'єднаних загальною метою і умовами їх виконання.

Примітка 1. Цілі на рівні програми, як правило, пов'язані із стратегічною метою організації. Уточнення цілей і вимог до результатів у міру просування програми є частим явищем.

Примітка 2. Виконання окремого проєкту у складі програми може не давати відчутного результату (доходу), тоді як здійснення всієї програми забезпечує максимальну ефективність (прибуток).

У більшості випадків програма може розглядатись, як складний, великий за обсягом та залученими ресурсами магaproєкт.

Програмне забезпечення для управління проєктами. (Project Management Software) – Клас комп'ютерних програм, розроблених спеціально для планування і контролю витрат, термінів і інших компонентів проєкту.

Продукт проєкту (project product) — основний кінцевий результат проєкту, який поставляється замовнику, звичайно визначений в проектному завданні (специфікації) на продукт.

Проект (Project) – Цілеспрямована діяльність тимчасового характеру, призначена для створення унікального продукту або послуги. Реалізації проєкту властиві специфічні способи організації робіт і управління.

Примітка 1. Проєкти направлені на досягнення конкретних цілей. Проте в деяких проєктах мети і вимоги до результатів можуть уточнювати міру виконання проєкту.

Примітка 2. Ступінь унікальності може значно відрізнятися від одного проєкту до іншого. Унікальність може бути пов'язана як з кінцевою метою проєкту, так і з умовами їх досягнення.

Примітка 3. Як правило, проєкти припускають необхідність координованого виконання взаємозв'язаних дій декількома виконавцями.

Примітка 4. Проект припускає створення тимчасової організаційної структури для досягнення поставленої мети.

Примітка 5. Крупний проєкт може складатися з декількох проєктів (підпроєктів).

Під-проєкти (subprojects). Проєкти часто підрозділяються на компоненти, для яких організовується виділене управління (під-проєкти). Під-проєкти можуть

виділятися відповідно до ходу проекту (такі як окремі фази проекту) або відповідно до структури і специфіки одержуваних результатів (продуктів) проекту. Під-проекти часто прирівнюються до проектів і управляються як проекти.

Проектно-орієнтована організація. (Projectized Organization) – Будь-яка організаційна структура, в якій менеджер проекту має повні повноваження для визначення пріоритетів і для керівництва роботою осіб, привернутих для роботи в рамках даного проекту.

Проектно-орієнтоване управління. (Management Projects) - Управлінський підхід, при якому багато замовлень і задачі виробничої діяльності організації, розглядаються як окремі проекти, до яких застосовуються принципи і методи управління проектами.

Проектний офіс (далі – ПО) – підрозділ Підприємства, на який покладаються обов’язки по організації управління проектами.

Проектні ризики – Невизначені події або можливі ситуації, що негативно впливають на досягнення успішних результатів проекту загалом і на окремі проектні результати або події, що може спричинити непередбачувані збитки. Проектні ризики визначаються ймовірністю появи ризикових подій і потенційними збитками.

Регламент – Нормативний документ, що описує послідовність операцій, відповідальність, порядок взаємодії виконавців (учасників), а також порядок використання ресурсів та процедуру управління і прийняття рішень щодо поліпшення ефективності бізнес-процесу управління проектами.

Реєстр проектів – Упорядкована сукупність проектів, портфелів та папок проектів, що опубліковані на сервері проектів (Project Server) КСУП. Для аналізу та управління реєстром проектів використовуються спеціальні програмні засоби.

Робота (Activity) – Елемент проекту, виконуваний в процесі його здійснення. Робота звичайно має очікувану (планову) тривалість (тривалість), передбачувану (очікувану) вартість і необхідні ресурси. Робота часто підрозділяється на завдання. На виконання роботи призначається відповідальний чи виконавець.

Розклад проекту. (Project Schedule) – Планові дати для виконання робіт і планові дати для настання контрольних (ключових) подій («віх») проекту.

Розклад контрольних подій. (Milestone Schedule) – Розклад узагальнюючого рівня, який відображає терміни настання контрольних подій і етапів.

Результат проекту. (Deliverable) – Будь-який вимірний, матеріальний, визначений «вихід» або результат, який повинен бути одержаний для завершення проекту або його частини. Звичайно використовується в більш вузькому значенні по

відношенню до зовнішніх результатів, які є об'єктом затвердження спонсором або клієнтом.

Спонсор проєкту – Відповідальний за певний проєкт керівник дирекції чи іншого структурного підрозділу товариства, який має достатні повноваження та ресурси для забезпечення виконання цього проєкту. Спонсор затверджує бюджет проєкту, рішення по змісту та терміни виконання проєкту. Спонсор має можливість зупинити виконання проєкту, змінити строки та певні задачі проєкту, надати, при необхідності, додаткові ресурси для виконання проєкту, як матеріальні, так і людські.

Стратегія проєкту (Project Strategy) визначає напрями і основні принципи здійснення проєкту; описує які проміжні результати і в якій послідовності повинні бути виконані для досягнення всієї сукупності стоять перед проєктом кінцевої мети.

Структурна декомпозиція робіт. (Work Breakdown Structure) – Ієрархічна структуризація робіт проєкту, орієнтована на основні результати проєкту, що визначають його наочну область. Кожний нижчестоящий рівень структури є деталізацією елементу вищого рівня проєкту. Елементом проєкту може бути як продукт, послуга так і пакет робіт або робота.

Структури проєкту. (Project Structures) – ієрархічні декомпозиції проєкту на складові частини (елементи, модулі), необхідні і достатні для ефективного здійснення процесу управління проєктом на користь різних учасників проєкту.

Управління інтеграційними процесами в проєкті. (Project Integration Management) – Розділ управління проєктами, який включає процеси, що потребуються для відповідної координації різних елементів проєкту. Сюди входять розробка плану проєкту, виконання плану проєкту і загальний контроль змін.

Управління якістю в проєкті. (Project Quality Management) – Розділ управління проєктами, який включає процеси, необхідні для забезпечення гарантій того, що проєкт задовольнить потребам, ради яких він і був зроблений. Включає планування якості, забезпечення якості і контроль якості.

Управління комунікаціями в проєкті. (Project Communications Management) – Розділ управління проєктами, що включає процеси, які необхідні для організації збору і розподілу достовірної проєктної інформації. Складається з планування комунікацій, розподілу інформації, наданні звітності про виконання проєкту і адміністративного завершення.

Управління контрактами і поставками в проєкті. (Project Procurement Management) – Розділ управління проєктами, який включає процеси, що потрібні для забезпечення поставки продуктів і послуг ззовні. Включає планування поставок,

планування пропозицій, запит пропозиції, вибір джерел, адміністрування контракту, закриття контракту.

Управління конфліктами в проєкті. (Project Conflict Management) – процес, в якому за допомогою використовування управлінських технологій вирішуються різні неузгодження та конфлікти, як технічного, так і особового характеру, які виникають у рамках роботи над проєктом.

Управління змістом проєкту. (Project Scope Management) – Розділ управління проєктами, який включає процеси, необхідні для забезпечення того, що в проєкт включені всі необхідні роботи і лише ті роботи, які необхідні для успішного завершення проєкту. Включає ініціацію робіт, планування наочної області, визначення наочної області, підтвердження наочної області і контроль змін наочної області.

Управління проєктом за часовими параметрами. (Project Time Management) – Розділ управління проєктами, який включає процеси управління проєктом за часовими параметрами, які необхідні для забезпечення своєчасного завершення проєкту, в т.ч.: визначення робіт, визначення послідовності робіт, оцінки тривалості робіт, розробка календарного плану і контроль календарного плану.

Управління проєктом. (Project Management (PM)) – Виконання дій, що використовують знання, навички, методи, засоби і технології при виконанні проєкту з метою досягнення або перевищення очікувань результатів проєкту. Планування, організація, моніторинг і контроль усіх складових проєкту, а також мотивація учасників проєкту для досягнення його цілей в обумовлений час з узгодженими вартістю та якістю виконання робіт.

Управління ризиками в проєкті. (Project Risk Management) – Розділ управління проєктами, що включає процеси, які пов'язані з визначенням, аналізом і відповідними заходами реагування на ризики в проєкті. Включає прогнозування і визначення ризиків, кількісну оцінку ризиків, розробку методів реагування на ризики і контроль реагування на ризики.

Управління вартістю проєкту. (Project Cost Management) – Розділ управління проєктами, що включає процеси, які необхідні для дотримання затвердженого бюджету проєкту. Складається з планування ресурсів, оцінки вартості, формування кошторису і бюджету і контролю вартості.

Управління людськими ресурсами проєкту. (Project Human Resource Management) – Розділ управління проєктами, який включає процеси, що вимагаються для найефективнішого використовування залученого до проєкту персоналу.

Складається з організаційного планування, підбору персоналу, створення і розвитку команди.

Учасники проєкту. (Stakeholders) – Фізичні особи і організації, які безпосередньо залучені в проєкт або чиї інтереси можуть бути зачеплені при здійсненні проєкту.

Фаза проєкту (Project Phase) – Набір логічно взаємозв'язаних робіт проєкту, в процесі завершення яких досягається один з основних або істотних проміжних результатів проєкту. Іноді можливе часткове поєднання або одночасне виконання окремих фаз проєкту.

Цілі проєкту. (Project Objectives) – Бажаний результат діяльності, що досягається в результаті успішного здійснення проєкту в заданих умовах його реалізації.

Члени команди проєкту. (Project Team Members) - Люди, які прямо чи побічно підзвітні менеджеру (керівнику) проєкту.

Додаток 2

Словник термінів Git та GitHub

Апстрим (upstream), коротка назва посилання на майстер-репозиторій.

Віддалений репозиторій - репозиторій, що знаходиться на віддаленому сервері. Це загальний репозиторій, до якого приходять усі зміни і з якого забираються усі оновлення.

Гілка (branch) – це паралельна версія репозиторію. Вона включена в цей репозиторій, але не впливає на головну версію, тим самим дозволяючи вільно працювати в паралельній. Коли ви внесли потрібні зміни, ви можете об'єднати їх з головною версією.

Гіт або Git – система контролю та керування версіями файлів.

Гітхаб або GitHub – веб-сервіс для розміщення репозиторіїв та спільноти розробки проектів.

Ішью (Issue), (*ср. р., не схиляється*) запит до організаторів репозиторію, що містить пропозицію щодо покращення, вказівку на помилку, завдання чи питання, пов'язані з репозиторієм, напр. *to open a new issue* – відкрити нове ішью.

Клонування (clone) — завантаження репозиторію з віддаленого сервера на локальний комп'ютер у певний каталог для подальшої роботи з цим каталогом як з репозиторієм.

Кодревью (code review), - процес перевірки коду на відповідність певним вимогам, завданням і зовнішньому вигляду.

Коміт (Commit) — фіксація змін або запис змін до репозиторію. **Commit** відбувається на локальній машині.

Контрибьютор (contributor), учасник проєкту з відкритим вихідним кодом, що допомагає його розвитку виправленням помилок, написанням коду та документації.

Локальний репозиторій – репозиторій, розташований на локальному комп'ютері розробника у каталозі. Саме в ньому відбувається розробка та фіксація змін, що видашують на віддалений репозиторій.

Майстер (Master) - головна або основна гілка репозиторію.

Майстер-репозиторій, головний репозиторій, від нього починаються форки.

Мердж (Merge) - злиття змін з будь-якої гілки репозиторію з будь-якою гілкою цього ж репозиторію. Найчастіше злиття змін із гілки репозиторію з основною гілкою репозиторію.

Мердж-реквест (merge request), синонім пул-реквесту, використовується в GitLab.

Мейнтейнер (maintainer) учасник проєкту з відкритим вихідним кодом, який приймає рішення щодо його розвитку та спрямовує хід розробки проєкту.

Оновитись з апстриму – оновити свою локальну версію форка до останньої версії основного репозиторію, від якого зроблено форк.

Оновитись з ориджину – оновити свою локальну версію репозиторію до останньої віддаленої версії цього репозиторію.

Ориджин (origin), коротка назва посилання на віддалений репозиторій.

Пул (Pull) – отримання останніх змін із віддаленого сервера репозиторію.

Пул-реквест (Pull Request) - запит на злиття форки (див. форк-репозиторій) репозиторію з основним репозиторієм. Пул-реквест може бути прийнятий або відхищений вами як власником репозиторію.

Пуш (Push) — надсилання всіх ненаправлених **commit**-ів на віддалений сервер репозиторію.

Репозиторій Git - каталог файлової системи, в якому знаходяться: файли конфігурації, файли журналів операцій, що виконуються над репозиторієм, індекс розташування файлів та сховище, що містить контролювані файли.

Сквошити або склеювати (squash), об'єднувати кілька **commit**-ів в один для чистішої історії змін: або вручну, або автоматично при мережі пул-реквеста.

Форк (Fork) – копія репозиторію. Його також можна як зовнішню гілку для поточного репозиторію. Копія вашого відкритого репозиторію на **Гітхабі** може бути зроблена будь-яким користувачем, після чого він може надіслати зміни до вашого репозиторію через пул-реквест.

Форк-репозиторій, форк майстер-репозиторію.