Тема 5. Методи планування проектів

Зміст

1. Вступ

Планування є одним із найважливіших етапів управління проектом розробки програмних систем. Це процес, який визначає, що потрібно зробити, хто повинен це зробити, скільки часу і ресурсів для цього знадобиться, і як оцінити результати. За статистикою, близько 70% проектів розробки програмного забезпечення стикаються з проблемами саме через недостатньо якісне планування.

Планування проекту — це безперервний процес, який починається на етапі ініціації і триває протягом усього життєвого циклу проекту. При цьому плани постійно уточнюються та адаптуються відповідно до змін у внутрішньому та зовнішньому середовищі проекту.

У проектах розробки програмних систем планування має низку особливостей, пов'язаних із нематеріальною природою програмного забезпечення, високою мінливістю вимог, складністю оцінки трудомісткості та залежністю від людського фактора.

У даній лекції ми розглянемо основні методи та інструменти планування проектів розробки програмних систем, зокрема планування часу, вартості, якості, розробку проектної документації, календарне планування, планування ресурсів та оцінку економічної ефективності проекту.

2. Планування часу проекту

2.1. Значення часового планування

Планування часу (або управління розкладом) є одним із ключових аспектів управління проектом, оскільки термін виконання часто є критичним обмеженням. Особливо це стосується проектів розробки програмних систем, де час виходу на ринок може визначати конкурентоспроможність продукту.

Основними цілями планування часу є:

- Встановлення реалістичних термінів виконання проекту
- Визначення логічної послідовності та взаємозв'язків між роботами
- Оптимізація розподілу ресурсів у часі
- Створення основи для моніторингу виконання проекту
- Забезпечення своєчасного завершення проекту

2.2. Процес планування часу

Процес планування часу проекту зазвичай включає наступні етапи:

1. **Визначення переліку робіт**:
    - На основі WBS (Work Breakdown Structure)
    - Визначення конкретних завдань та їх обсягу
    - Встановлення критеріїв завершення робіт
2. **Визначення взаємозв'язків між роботами**:
    - Логічні залежності між завданнями

    - Типи зв'язків: "Закінчення-Початок", "Початок-Початок", "Закінчення-Закінчення", "Початок-Закінчення"
    - Обмеження та припущення щодо послідовності робіт
3. **Оцінка тривалості робіт**:
    - Експертні оцінки
    - Аналогове оцінювання (на основі досвіду попередніх проектів)
    - Параметричне оцінювання (на основі кількісних показників)
    - Метод PERT (Program Evaluation and Review Technique)
    - Метод трьох точок (оптимістична, найбільш ймовірна, песимістична оцінки)
4. **Розробка розкладу**:
    - Метод критичного шляху (Critical Path Method, CPM)
    - Метод критичного ланцюга (Critical Chain Method, CCM)
    - Оптимізація ресурсів
    - Вирівнювання ресурсів
    - Аналіз сценаріїв "що-якщо"
5. **Контроль розкладу**:
    - Відстеження фактичного виконання
    - Аналіз відхилень
    - Оновлення та коригування розкладу
    - Управління змінами розкладу

2.3. Методи оцінки тривалості робіт

У проектах розробки програмного забезпечення оцінка тривалості робіт є одним з найскладніших аспектів планування через високу невизначеність та залежність від людського фактора. Розглянемо основні методи, які використовуються для оцінки тривалості робіт:

2.3.1. Експертна оцінка

Експертна оцінка базується на досвіді та знаннях фахівців, які мають досвід виконання подібних робіт. Переваги:

- Враховує специфіку конкретного проекту
- Легка в реалізації
- Не потребує значних витрат

Недоліки:

- Суб'єктивність
- Залежність від досвіду експертів
- Можливий оптимістичний або песимістичний ухил

2.3.2. Метод трьох точок (PERT)

Метод PERT (Program Evaluation and Review Technique) передбачає оцінку тривалості роботи на основі трьох оцінок:

- O (Optimistic) - оптимістична оцінка (мінімальний час)
- M (Most likely) - найбільш ймовірна оцінка
- P (Pessimistic) - песимістична оцінка (максимальний час)

Очікувана тривалість розраховується за формулою:

E = (O + 4M + P) / 6

Стандартне відхилення:

SD = (P - O) / 6

Цей метод дозволяє врахувати невизначеність та ризики при оцінці тривалості робіт.

2.3.3. Аналогове оцінювання

Аналогове оцінювання базується на досвіді виконання подібних робіт у минулих проектах. Застосовується, коли поточний проект має схожість з минулими проектами.

Етапи аналогового оцінювання:

1. Ідентифікація аналогічних робіт у минулих проектах

2. Аналіз фактичної тривалості цих робіт
3. Корегування оцінок з урахуванням відмінностей поточного проекту

2.3.4. Параметричне оцінювання

Параметричне оцінювання базується на використанні математичних моделей, які пов'язують тривалість робіт з певними параметрами (наприклад, кількістю рядків коду, функціональних точок тощо).

Наприклад, якщо відомо, що розробка одного модуля середньої складності займає 40 годин, а в системі планується 10 таких модулів, то загальна оцінка становитиме 400 годин.

2.3.5. Planning Poker (для Agile-проектів)

Planning Poker — це техніка оцінки, яка часто використовується в Agile-проектах. Члени команди одночасно показують карти з оцінками (зазвичай у story points), після чого обговорюють розбіжності та приходять до консенсусу.

Переваги:

- Залучення всієї команди
- Зменшення впливу авторитетів
- Стимулювання обговорення та обміну знаннями

2.4. Методи розробки розкладу

2.4.1. Метод критичного шляху (CPM)

Метод критичного шляху — це техніка аналізу мережі, яка визначає найдовший шлях у проекті від початку до завершення. Роботи, які лежать на критичному шляху, не мають резерву часу, і затримка в їх виконанні призведе до затримки всього проекту.

Основні етапи:

1. Побудова мережевої діаграми проекту
2. Визначення ранніх та пізніх термінів початку та завершення кожної роботи
3. Розрахунок резервів часу для кожної роботи
4. Визначення критичного шляху (роботи з нульовим резервом)

Критичний шлях визначає мінімальну тривалість проекту та допомагає зосередити увагу на найбільш критичних роботах.

2.4.2. Метод критичного ланцюга (CCM)

Метод критичного ланцюга — це модифікація методу критичного шляху, яка враховує обмеження ресурсів і поведінкові аспекти планування.

Особливості методу:

- Визначення критичного ланцюга з урахуванням ресурсних обмежень
- Використання буферів проекту та буферів живлення для управління невизначеністю
- Зменшення тривалості окремих завдань (видалення "страхових" запасів)
- Фокус на управлінні буферами, а не на виконанні конкретних дат

2.4.3. Метод хвилі, що котиться (Rolling Wave Planning)

У проектах з високим рівнем невизначеності (що характерно для розробки програмних систем) часто використовується метод "хвилі, що котиться". Цей метод передбачає детальне планування найближчих етапів і загальне планування віддалених етапів з поступовим уточненням по мірі наближення до них.

Переваги:

- Адаптивність до змін
- Ефективне використання часу планування
- Поступове зниження невизначеності

2.4.4. Гнучке планування (Agile Planning)

У Agile-методологіях використовується інкрементальний та ітеративний підхід до планування:

- **Релізне планування**: визначення загальних термінів і обсягу релізів

- **Планування спринту**: детальне планування робіт на найближчий спринт (зазвичай 2-4 тижні)
- **Щоденне планування**: короткі щоденні зустрічі для синхронізації та коригування планів

Особливості Agile-планування:

- Фокус на бізнес-цінності
- Адаптивність до змін
- Постійний зворотний зв'язок
- Емпіричний підхід (inspect and adapt)

2.5. Інструменти планування часу

Для планування часу проекту використовуються різноманітні інструменти:

1. **Діаграма Ганта** — горизонтальна діаграма, що відображає роботи проекту у вигляді відрізків на шкалі часу, показуючи початок, тривалість та кінець кожної роботи, а також взаємозв'язки між роботами.
2. **Мережеві діаграми (PERT, PDM)** — графічне представлення робіт проекту та їх взаємозв'язків, які допомагають визначити критичний шлях.
3. **Програмні засоби** — спеціалізоване ПЗ для управління проектами:
    - Microsoft Project
    - Oracle Primavera
    - Jira з плагінами для планування
    - Trello, Asana, Monday.com
    - Спеціалізовані інструменти для Agile (Rally, Version One)
4. **Канбан-дошки** — візуальний інструмент для відстеження прогресу робіт, особливо популярний в Agile-методологіях.

3. Планування вартості проекту

3.1. Значення планування вартості

Планування вартості (або управління бюджетом) є критично важливим аспектом управління проектом, оскільки визначає фінансові рамки проекту та впливає на його життєздатність.

Основними цілями планування вартості є:

- Визначення загального бюджету проекту
- Розподіл коштів за елементами робіт та періодами часу
- Забезпечення ефективного використання ресурсів
- Створення основи для моніторингу та контролю витрат
- Забезпечення фінансової життєздатності проекту

3.2. Процес планування вартості

Процес планування вартості проекту зазвичай включає наступні етапи:

1. **Оцінка вартості** — визначення приблизної вартості ресурсів, необхідних для виконання робіт проекту.
2. **Визначення бюджету** — агрегування оцінених вартостей окремих робіт для встановлення загального бюджету проекту та його розподілу за елементами WBS та періодами часу.
3. **Управління вартістю** — процес моніторингу статусу проекту для оновлення бюджету та управління змінами базового плану вартості.

3.3. Методи оцінки вартості

3.3.1. Експертна оцінка

Експертна оцінка базується на досвіді та знаннях фахівців, які мають досвід виконання подібних проектів. Аналогічно до оцінки часу, експертна оцінка вартості може бути суб'єктивною, але часто є єдиним доступним методом для унікальних проектів.

3.3.2. Аналогове оцінювання

Аналогове оцінювання використовує фактичну вартість подібних проектів у минулому як основу для оцінки поточного проекту. Цей метод особливо корисний на ранніх етапах планування, коли детальна інформація ще недоступна.

3.3.3. Параметричне оцінювання

Параметричне оцінювання використовує статистичні взаємозв'язки між історичними даними та іншими змінними для оцінки вартості. Наприклад, вартість розробки програмного забезпечення може оцінюватися на основі кількості функціональних точок або рядків коду.

3.3.4. Оцінка "знизу-вгору" (Bottom-up Estimating)

Цей метод передбачає оцінку вартості окремих робіт на найнижчому рівні WBS з подальшим агрегуванням для отримання загальної вартості проекту. Цей метод зазвичай дає найбільш точні оцінки, але вимагає детального планування.

3.3.5. Оцінка за трьома точками

Аналогічно до оцінки часу, оцінка вартості може базуватися на трьох сценаріях:

- Оптимістична оцінка (найнижча вартість)
- Найбільш ймовірна оцінка
- Песимістична оцінка (найвища вартість)

Очікувана вартість розраховується за формулою:

E = (O + 4M + P) / 6

3.3.6. Аналіз резервів

Для врахування невизначеності та ризиків у бюджеті проекту використовуються резерви:

- **Резерв на відомі ризики (Contingency Reserve)** — кошти, виділені для покриття ідентифікованих ризиків з певною ймовірністю виникнення.
- **Резерв на невідомі ризики (Management Reserve)** — кошти, виділені для покриття неідентифікованих ризиків або непередбачених змін обсягу проекту.

3.4. Структура витрат проекту розробки програмної системи

У проектах розробки програмних систем витрати зазвичай структуруються наступним чином:

1. **Витрати на персонал**:
    - Заробітна плата команди розробників
    - Заробітна плата менеджерів та адміністративного персоналу
    - Бонуси та премії
    - Соціальні пакети та страхування
    - Навчання та підвищення кваліфікації
2. **Обладнання та інфраструктура**:
    - Комп'ютерне обладнання
    - Серверне обладнання
    - Мережеве обладнання
    - Офісне обладнання
    - Амортизація обладнання
3. **Програмне забезпечення**:
    - Ліцензії на операційні системи
    - Ліцензії на середовища розробки (IDE)
    - Ліцензії на СУБД
    - Ліцензії на інструменти тестування
    - Ліцензії на інструменти управління проектами
4. **Хмарні сервіси та хостинг**:
    - Оплата хмарних обчислювальних ресурсів
    - Оплата хмарних сховищ даних
    - Оплата хостингу
    - Оплата доменних імен

    - Оплата CDN-сервісів
5. **Адміністративні витрати**:
    - Оренда офісних приміщень
    - Комунальні послуги
    - Телекомунікаційні послуги
    - Канцелярські товари
    - Маркетинг та просування
6. **Зовнішні послуги**:
    - Консультанти та експерти
    - Аутсорсинг окремих робіт
    - Юридичні послуги
    - Бухгалтерські послуги
    - Аудит та сертифікація
7. **Резерви**:
    - Резерв на відомі ризики
    - Резерв на невідомі ризики

3.5. Методи контролю вартості

Контроль вартості проекту здійснюється з використанням різних методів та інструментів:

3.5.1. Метод освоєного обсягу (Earned Value Management, EVM)

Метод освоєного обсягу — це техніка, яка поєднує аналіз обсягу, графіка та вартості для оцінки прогресу та продуктивності проекту.

Основні показники EVM:

- **PV (Planned Value)** — плановий обсяг робіт, який повинен бути виконаний до певного моменту часу.
- **EV (Earned Value)** — фактично виконаний обсяг робіт, виражений у планових витратах.
- **AC (Actual Cost)** — фактичні витрати на виконання робіт.

На основі цих показників розраховуються індекси та відхилення:

- **CV (Cost Variance)** = EV - AC (відхилення за вартістю)
- **SV (Schedule Variance)** = EV - PV (відхилення за графіком)
- **CPI (Cost Performance Index)** = EV / AC (індекс виконання вартості)
- **SPI (Schedule Performance Index)** = EV / PV (індекс виконання графіка)

Ці показники дозволяють оцінити ефективність використання ресурсів та прогнозувати кінцеву вартість проекту.

3.5.2. Аналіз відхилень

Аналіз відхилень передбачає порівняння фактичних витрат з плановими для виявлення розбіжностей та їх причин. Виявлені відхилення аналізуються для визначення необхідних коригувальних дій.

3.5.3. Прогнозування

На основі даних про фактичні витрати та прогрес проекту здійснюється прогнозування кінцевої вартості проекту. Основні показники прогнозування:

- **EAC (Estimate at Completion)** — прогноз загальної вартості проекту при завершенні.
- **ETC (Estimate to Complete)** — прогноз витрат, необхідних для завершення проекту.
- **VAC (Variance at Completion)** — відхилення загальної вартості проекту від бюджету.

3.6. Бюджетування проекту

Бюджетування — це процес розподілу загальної вартості проекту за періодами часу та елементами WBS для створення базового плану вартості.

Основні методи бюджетування:

1. **Бюджетування "зверху-вниз"** — розподіл загального бюджету за елементами WBS на основі їх відносної важливості або обсягу робіт.
2. **Бюджетування "знизу-вгору"** — агрегування оцінок вартості окремих робіт для формування загального бюджету.
3. **Параметричне бюджетування** — розподіл бюджету на основі кількісних параметрів (наприклад, кількості функціональних точок, рядків коду тощо).
4. **Бюджетування за аналогією** — використання даних минулих проектів для розподілу бюджету поточного проекту.

4. Планування якості проекту

4.1. Значення планування якості

Планування якості — це процес визначення вимог до якості продукту та проекту, а також встановлення способів дотримання цих вимог. У проектах розробки програмних систем якість має особливе значення, оскільки впливає на надійність, безпеку, продуктивність та зручність використання системи.

Основними цілями планування якості є:

- Визначення стандартів якості, які застосовуються до проекту
- Визначення метрик та індикаторів якості
- Визначення процесів забезпечення якості
- Визначення процесів контролю якості
- Мінімізація витрат на виправлення дефектів

4.2. Процес планування якості

Процес планування якості включає наступні етапи:

1. **Визначення вимог до якості**:
    - Ідентифікація стандартів та нормативів
    - Аналіз очікувань зацікавлених сторін
    - Визначення критеріїв приймання продукту
    - Визначення критичних факторів якості
2. **Розробка плану управління якістю**:
    - Визначення підходу до забезпечення якості
    - Визначення ролей та відповідальності за якість
    - Визначення процесів та процедур забезпечення якості
    - Визначення необхідних ресурсів
3. **Визначення метрик якості**:
    - Вибір показників для вимірювання якості
    - Встановлення цільових значень показників
    - Визначення методів збору та аналізу даних
4. **Розробка плану поліпшення процесів**:
    - Аналіз поточних процесів
    - Визначення областей для поліпшення
    - Розробка заходів з поліпшення
    - Встановлення механізмів моніторингу та контролю

4.3. Стандарти якості програмного забезпечення

У проектах розробки програмних систем використовуються різноманітні стандарти якості:

1. **ISO/IEC 25010:2011** — міжнародний стандарт, який визначає модель якості програмного забезпечення та системи. Включає вісім характеристик якості:
    - Функціональна придатність
    - Ефективність
    - Сумісність
    - Зручність використання
    - Надійність

    - Безпека
    - Супроводжуваність
    - Переносимість
2. **ISO/IEC 12207:2017** — міжнародний стандарт, який визначає процеси життєвого циклу програмного забезпечення.
3. **CMMI (Capability Maturity Model Integration)** — модель зрілості процесів, яка визначає рівні зрілості організації в розробці програмного забезпечення.
4. **IEEE 730-2014** — стандарт для планів забезпечення якості програмного забезпечення.
5. **IEEE 1012-2016** — стандарт для верифікації та валідації програмного забезпечення.

4.4. Методи забезпечення якості

4.4.1. Рецензування коду (Code Review)

Рецензування коду — це систематичний аналіз коду іншими розробниками для виявлення помилок, недоліків та можливостей для поліпшення. Рецензування може бути формальним (з використанням чек-листів та документуванням результатів) або неформальним (парне програмування, групове обговорення).

4.4.2. Статичний аналіз коду

Статичний аналіз коду — це автоматизований аналіз коду без його виконання для виявлення потенційних дефектів, порушень стандартів кодування, вразливостей безпеки тощо. Для статичного аналізу використовуються спеціалізовані інструменти (SonarQube, ESLint, Checkstyle, PMD тощо).

4.4.3. Тестування

Тестування — це процес виконання програми з метою виявлення дефектів. У проектах розробки програмних систем використовуються різні види тестування:

- **Модульне тестування (Unit Testing)** — тестування окремих модулів або компонентів програми.
- **Інтеграційне тестування (Integration Testing)** — тестування взаємодії між компонентами програми.
- **Системне тестування (System Testing)** — тестування системи в цілому на відповідність вимогам.
- **Приймальне тестування (Acceptance Testing)** — тестування системи на відповідність потребам користувачів.

4.4.4. Автоматизоване тестування

Автоматизоване тестування — це використання спеціального програмного забезпечення для виконання тестів та порівняння фактичних результатів з очікуваними. Автоматизація тестування дозволяє збільшити покриття, прискорити виконання тестів та забезпечити регресійне тестування.

4.4.5. Безперервна інтеграція та безперервна доставка (CI/CD)

CI/CD — це практика автоматизації процесів збірки, тестування та розгортання програмного забезпечення. CI/CD забезпечує швидкий зворотний зв'язок щодо якості коду, зменшує ризики інтеграції та прискорює виправлення дефектів.

4.5. Метрики якості програмного забезпечення

Для оцінки якості програмного забезпечення використовуються різноманітні метрики:

1. **Метрики коду**:
    - Цикломатична складність
    - Покриття коду тестами
    - Дублювання коду
    - Кількість рядків коду
    - Кількість коментарів
2. **Метрики дефектів**:
    - Щільність дефектів (кількість дефектів на 1000 рядків коду)

    - Час до виявлення дефекту
    - Час до виправлення дефекту
    - Частота відмов
    - Середній час між відмовами (MTBF)
3. **Метрики продуктивності**:
    - Час відгуку
    - Пропускна здатність
    - Використання ресурсів
    - Масштабованість
    - Ефективність
4. **Метрики зручності використання**:
    - Задоволеність користувачів
    - Час виконання типових завдань
    - Кількість помилок користувачів
    - Легкість навчання

4.6. Планування тестування

Планування тестування є важливою частиною планування якості. План тестування повинен включати:

1. **Стратегію тестування**:
    - Види тестування, які будуть проводитись
    - Підходи до тестування (ручне, автоматизоване)
    - Критерії входу та виходу для тестування
2. **Ресурси для тестування**:
    - Людські ресурси (тестувальники, експерти)
    - Технічні ресурси (обладнання, середовища)
    - Інструменти тестування
3. **Графік тестування**:
    - Терміни проведення тестування
    - Залежності від інших робіт
    - Віхи тестування
4. **Тестові артефакти**:
    - Тестові вимоги
    - Тестові сценарії
    - Тестові випадки
    - Тестові дані
5. **Процедури тестування**:
    - Процес виконання тестів
    - Процедури реєстрації та відстеження дефектів
    - Процедури звітності
6. **Критерії завершення тестування**:
    - Критерії успішного завершення тестування
    - Критерії прийнятності результатів
    - Умови для припинення тестування

5. Розробка документації по проекту

5.1. Значення проектної документації

Проектна документація — це сукупність текстових та графічних документів, які описують цілі, обсяг, вимоги, обмеження, рішення та результати проекту. Якісна документація забезпечує:

- Чітке розуміння цілей та обсягу проекту всіма зацікавленими сторонами
- Ефективну комунікацію між учасниками проекту
- Збереження та передачу знань та досвіду

- Можливість відстеження прогресу та змін
- Основу для прийняття рішень
- Можливість аудиту та оцінки проекту

5.2. Види проектної документації

У проектах розробки програмних систем зазвичай розробляються наступні види документації:

1. **Ініціювальна документація**:
    - Статут проекту (Project Charter)
    - Документ про ініціацію проекту (Project Initiation Document)
    - Техніко-економічне обґрунтування (Business Case)
2. **Планувальна документація**:
    - План управління проектом (Project Management Plan)
    - План управління обсягом (Scope Management Plan)
    - План управління розкладом (Schedule Management Plan)
    - План управління вартістю (Cost Management Plan)
    - План управління якістю (Quality Management Plan)
    - План управління ресурсами (Resource Management Plan)
    - План управління комунікаціями (Communications Management Plan)
    - План управління ризиками (Risk Management Plan)
    - План управління закупівлями (Procurement Management Plan)
    - План управління зацікавленими сторонами (Stakeholder Management Plan)
3. **Технічна документація**:
    - Специфікація вимог до програмного забезпечення (Software Requirements Specification)
    - Документ архітектури програмного забезпечення (Software Architecture Document)
    - Документ проектування програмного забезпечення (Software Design Document)
    - Технічне завдання (Technical Assignment)
    - Інтерфейсні специфікації (Interface Specifications)
4. **Тестова документація**:
    - План тестування (Test Plan)
    - Тестові сценарії (Test Scenarios)
    - Тестові випадки (Test Cases)
    - Звіти про тестування (Test Reports)
    - Звіти про дефекти (Defect Reports)
5. **Користувацька документація**:
    - Керівництво користувача (User Guide)
    - Керівництво адміністратора (Administrator Guide)
    - Навчальні матеріали (Training Materials)
    - Довідкова документація (Reference Documentation)
6. **Документація з експлуатації та підтримки**:
    - Керівництво з експлуатації (Operations Manual)
    - Керівництво з технічного обслуговування (Maintenance Manual)
    - Керівництво з усунення несправностей (Troubleshooting Guide)
7. **Звітна документація**:
    - Звіти про стан проекту (Project Status Reports)
    - Звіти про виконання робіт (Work Performance Reports)
    - Запити на зміни (Change Requests)
    - Протоколи зустрічей (Meeting Minutes)
    - Підсумковий звіт проекту (Project Closure Report)

5.3. Процес розробки документації

Процес розробки документації включає наступні етапи:

1. **Планування документації**:
    - Визначення необхідних документів
    - Визначення аудиторії та цілей
    - Встановлення стандартів та шаблонів
    - Визначення відповідальних за розробку
2. **Розробка документації**:
    - Збір інформації
    - Написання та редагування текстів
    - Створення графічних матеріалів
    - Рецензування та затвердження
3. **Управління документацією**:
    - Зберігання та контроль версій
    - Розповсюдження та доступ
    - Оновлення та підтримка
    - Архівування та знищення

5.4. Методи розробки документації

5.4.1. Традиційний підхід

Традиційний підхід передбачає розробку повного набору документації до початку реалізації проекту. Цей підхід характерний для водоспадної (Waterfall) методології та вимагає значних зусиль на початкових етапах проекту.

Переваги:

- Повне визначення проекту до початку реалізації
- Чіткі критерії для оцінки прогресу
- Зменшення ризиків, пов'язаних з неправильним розумінням вимог

Недоліки:

- Висока чутливість до змін
- Затримка початку реалізації
- Можливість застарівання документації
- Великий обсяг документації, який складно підтримувати

5.4.2. Гнучкий підхід (Agile Documentation)

Гнучкий підхід передбачає розробку документації інкрементально, паралельно з розробкою програмного забезпечення. Цей підхід характерний для Agile-методологій та фокусується на "достатній" документації.

Принципи Agile-документації:

- Створення документації, яка має реальну цінність для зацікавлених сторін
- Розробка документації "точно вчасно" (Just-in-Time)
- Використання легких форматів (wiki, markdown, діаграми)
- Фокус на колаборації та спільному розумінні, а не на формальних документах
- Регулярне оновлення та перевірка актуальності

5.4.3. Документація як код (Documentation as Code)

Підхід "документація як код" передбачає застосування принципів розробки програмного забезпечення до розробки документації. Документація зберігається в системі контролю версій, проходить рецензування та автоматично збирається та публікується.

Переваги:

- Можливість використання тих самих інструментів, що і для розробки коду
- Автоматизація процесів збірки та публікації
- Відстеження змін та історії
- Спрощення колаборації та рецензування

5.5. Інструменти для розробки документації

Для розробки проектної документації використовуються різноманітні інструменти:

1. **Текстові редактори та процесори**:
    - Microsoft Word
    - Google Docs
    - LibreOffice Writer
2. **Інструменти для управління вимогами**:
    - Jira
    - Confluence
    - IBM Rational DOORS
    - ReqView
3. **Інструменти для моделювання**:
    - Enterprise Architect
    - Visual Paradigm
    - Lucidchart
    - Draw.io
4. **Системи для документації як коду**:
    - Markdown з Git
    - AsciiDoc
    - Sphinx
    - Jekyll
    - GitBook
5. **Інструменти для API-документації**:
    - Swagger / OpenAPI
    - Postman
    - API Blueprint
    - ReadTheDocs

6. Календарне планування

6.1. Суть та значення календарного планування

Календарне планування — це процес визначення термінів виконання робіт проекту та створення графіка, який показує, коли кожна робота повинна початися та закінчитися.
Календарне планування забезпечує:

- Визначення логічної послідовності та тривалості робіт
- Оптимізацію використання ресурсів
- Координацію діяльності різних учасників проекту
- Основу для моніторингу та контролю прогресу проекту
- Можливість раннього виявлення потенційних проблем

6.2. Методи календарного планування

6.2.1. Метод діаграм Ганта

Діаграма Ганта — це горизонтальна діаграма, яка відображає роботи проекту у вигляді відрізків на шкалі часу. Діаграма Ганта показує:

- Початок та кінець кожної роботи
- Тривалість кожної роботи
- Взаємозв'язки між роботами
- Прогрес виконання кожної роботи
- Віхи проекту

Переваги:

- Наочність та простота розуміння
- Можливість відстеження прогресу
- Широка підтримка програмними засобами

Недоліки:

- Обмежені можливості відображення складних залежностей

- Складність відображення великих проектів
- Недостатня увага до ресурсних обмежень

6.2.2. Метод мережевого планування (PERT/CPM)

Метод мережевого планування базується на побудові мережевої діаграми проекту, яка відображає логічні взаємозв'язки між роботами. Два основних методи мережевого планування:

- **PERT (Program Evaluation and Review Technique)** — метод, який використовує оцінки тривалості робіт за трьома сценаріями (оптимістичний, найбільш ймовірний, песимістичний) для розрахунку очікуваної тривалості та стандартного відхилення.
- **CPM (Critical Path Method)** — метод, який визначає критичний шлях проекту — послідовність робіт, які визначають загальну тривалість проекту.

Переваги:

- Можливість аналізу критичного шляху
- Врахування залежностей між роботами
- Можливість аналізу резервів часу
- Можливість аналізу ризиків за тривалістю

Недоліки:

- Складність побудови та аналізу для великих проектів
- Необхідність спеціалізованого програмного забезпечення
- Фокус на часових, а не на ресурсних обмеженнях

6.2.3. Метод критичного ланцюга (CCPM)

Метод критичного ланцюга (Critical Chain Project Management) — це модифікація методу критичного шляху, яка враховує ресурсні обмеження та поведінкові аспекти управління проектами.

Особливості методу:

- Визначення критичного ланцюга з урахуванням ресурсних залежностей
- Використання буферів проекту для управління невизначеністю
- Фокус на доступності ресурсів
- Зменшення "страхових" запасів часу в оцінках тривалості робіт

6.2.4. Гнучке календарне планування (Agile Scheduling)

У Agile-проектах календарне планування здійснюється інкрементально та ітеративно:

- **Релізне планування** — визначення загальних термінів та змісту релізів
- **Планування спринту** — детальне планування робіт на найближчий спринт (2-4 тижні)
- **Щоденне планування** — коригування планів на основі щоденних обговорень

Особливості:

- Фокус на створенні цінності для користувача
- Адаптивність до змін
- Короткі цикли зворотного зв'язку
- Емпіричний підхід до планування

6.3. Розробка календарного плану

Процес розробки календарного плану включає наступні етапи:

1. **Визначення переліку робіт** (на основі WBS):
    - Декомпозиція проекту на окремі роботи
    - Визначення обсягу кожної роботи
    - Визначення критеріїв завершення кожної роботи
2. **Визначення взаємозв'язків між роботами**:
    - Логічні залежності (Finish-to-Start, Start-to-Start, Finish-to-Finish, Start-to-Finish)
    - Технологічні обмеження

    - Ресурсні залежності
    - Зовнішні залежності
3. **Оцінка тривалості робіт**:
    - Експертна оцінка
    - Аналогове оцінювання
    - Параметричне оцінювання
    - Оцінка за трьома точками
4. **Розробка мережевої діаграми**:
    - Побудова логічної послідовності робіт
    - Визначення критичного шляху
    - Розрахунок резервів часу
5. **Призначення ресурсів**:
    - Визначення необхідних ресурсів для кожної роботи
    - Перевірка доступності ресурсів
    - Вирівнювання ресурсів за необхідності
6. **Оптимізація календарного плану**:
    - Стиснення розкладу (crashing) — додавання ресурсів для скорочення тривалості
    - Швидкий прохід (fast tracking) — виконання робіт паралельно
    - Вирівнювання ресурсів — розподіл навантаження на ресурси
    - Аналіз сценаріїв "що-якщо"
7. **Затвердження базового календарного плану**:
    - Узгодження з ключовими зацікавленими сторонами
    - Фіксація базового плану для подальшого відстеження
    - Документування припущень та обмежень

6.4. Управління календарним планом

Управління календарним планом включає моніторинг виконання плану, аналіз відхилень та коригування за необхідності:

1. **Моніторинг прогресу**:
    - Відстеження фактичного виконання робіт
    - Порівняння фактичного прогресу з плановим
    - Оновлення статусу робіт
2. **Аналіз відхилень**:
    - Виявлення відхилень від плану
    - Аналіз причин відхилень
    - Оцінка впливу відхилень на загальний графік проекту
3. **Прогнозування**:
    - Прогнозування термінів завершення проекту
    - Прогнозування термінів досягнення ключових віх
    - Ідентифікація потенційних проблем
4. **Коригувальні дії**:
    - Перепланування робіт
    - Перерозподіл ресурсів
    - Стиснення графіка
    - Перегляд обсягу проекту
5. **Управління змінами**:
    - Оцінка впливу змін на календарний план
    - Затвердження або відхилення змін
    - Оновлення базового плану за необхідності

6.5. Інструменти календарного планування

Для календарного планування використовуються різноманітні інструменти:

1. **Спеціалізоване програмне забезпечення**:
    - Microsoft Project
    - Oracle Primavera
    - OpenProject
    - GanttProject
2. **Agile-інструменти**:
    - Jira
    - Trello
    - Asana
    - Monday.com
3. **Канбан-дошки**:
    - Фізичні дошки з картками
    - Електронні канбан-дошки (Jira, Trello)
4. **Інструменти для спільної роботи**:
    - Microsoft Teams
    - Slack
    - Google Workspace
    - Basecamp

7. Планування ресурсів

7.1. Поняття та види ресурсів проекту

Ресурси проекту — це все, що необхідно для виконання робіт проекту. У проектах розробки програмних систем виділяють наступні види ресурсів:

1. **Людські ресурси**:
    - Розробники
    - Тестувальники
    - Аналітики
    - Дизайнери
    - Менеджери
    - Технічні експерти
2. **Матеріальні ресурси**:
    - Комп'ютерне обладнання
    - Серверне обладнання
    - Мережеве обладнання
    - Офісне обладнання
    - Офісні приміщення
3. **Нематеріальні ресурси**:
    - Програмне забезпечення
    - Ліцензії
    - Патенти
    - Інтелектуальна власність
    - Бази даних
4. **Фінансові ресурси**:
    - Бюджет проекту
    - Інвестиції
    - Оборотні кошти
    - Резерви

7.2. Процес планування ресурсів

Процес планування ресурсів включає наступні етапи:

1. **Ідентифікація необхідних ресурсів**:
    - Визначення типів та кількості ресурсів для кожної роботи
    - Визначення вимог до ресурсів (кваліфікація, досвід, продуктивність)

    - Визначення термінів використання ресурсів
2. **Оцінка доступності ресурсів**:
    - Аналіз наявних ресурсів організації
    - Оцінка можливості залучення зовнішніх ресурсів
    - Аналіз конфліктів ресурсів між проектами
3. **Розподіл ресурсів**:
    - Призначення ресурсів для кожної роботи
    - Оптимізація використання ресурсів
    - Вирівнювання навантаження на ресурси
4. **Планування придбання ресурсів**:
    - Визначення термінів та способів придбання ресурсів
    - Планування процесів закупівлі
    - Планування контрактів з постачальниками
5. **Розробка плану управління ресурсами**:
    - Визначення підходу до управління ресурсами
    - Визначення ролей та відповідальності
    - Визначення процедур звітності та контролю

7.3. Планування людських ресурсів

Планування людських ресурсів є особливо важливим у проектах розробки програмних систем, оскільки люди є ключовим ресурсом таких проектів.

7.3.1. Визначення ролей та відповідальності

Для кожної ролі в проекті визначаються:

- Функціональні обов'язки
- Повноваження
- Відповідальність
- Кваліфікаційні вимоги
- Ключові показники ефективності (KPI)

Для визначення та документування ролей та відповідальності використовуються різні інструменти:

- Матриця відповідальності (RACI)
- Організаційні діаграми
- Посадові інструкції
- Робочі пакети WBS

7.3.2. Комплектування команди

Комплектування команди проекту включає:

1. **Визначення необхідного складу команди**:
    - Аналіз вимог проекту
    - Визначення необхідних ролей
    - Визначення оптимального розміру команди
    - Визначення структури команди
2. **Пошук та підбір персоналу**:
    - Внутрішній рекрутинг (залучення співробітників організації)
    - Зовнішній рекрутинг (найм нових співробітників)
    - Аутсорсинг (залучення зовнішніх підрядників)
    - Аутстафінг (залучення персоналу через посередника)
3. **Формування команди**:
    - Визначення структури команди
    - Визначення ролей та відповідальності
    - Встановлення правил взаємодії
    - Проведення командоутворюючих заходів

7.3.3. Планування розвитку команди

Планування розвитку команди включає:

1. **Оцінка компетенцій**:
    - Визначення наявних компетенцій
    - Визначення необхідних компетенцій
    - Аналіз розривів
2. **Розробка плану навчання та розвитку**:
    - Визначення форм та методів навчання
    - Планування тренінгів та навчальних заходів
    - Планування наставництва та коучингу
    - Планування самоосвіти та саморозвитку
3. **Планування кар'єрного розвитку**:
    - Визначення кар'єрних траєкторій
    - Планування ротації персоналу
    - Планування підвищення кваліфікації
    - Планування сертифікації

7.4. Планування матеріальних та нематеріальних ресурсів

Планування матеріальних та нематеріальних ресурсів включає:

1. **Визначення потреб**:
    - Аналіз вимог проекту
    - Визначення необхідних ресурсів
    - Визначення характеристик та параметрів ресурсів
    - Визначення термінів використання ресурсів
2. **Аналіз можливостей**:
    - Аналіз наявних ресурсів
    - Аналіз можливості повторного використання
    - Аналіз можливості оренди або лізингу
    - Аналіз можливості придбання
3. **Розробка плану придбання**:
    - Визначення способів придбання (купівля, оренда, лізинг)
    - Визначення термінів придбання
    - Визначення бюджету придбання
    - Визначення процедур придбання
4. **Планування використання**:
    - Розподіл ресурсів за роботами
    - Оптимізація використання ресурсів
    - Планування обслуговування та підтримки
    - Планування утилізації

7.5. Методи оптимізації використання ресурсів

7.5.1. Вирівнювання ресурсів (Resource Leveling)

Вирівнювання ресурсів — це техніка, яка використовується для усунення перевантаження ресурсів шляхом зміни термінів виконання робіт у межах наявного резерву часу.

Методи вирівнювання ресурсів:

- Зміщення початку робіт
- Розтягування тривалості робіт
- Переривання робіт
- Зміна послідовності робіт

7.5.2. Згладжування ресурсів (Resource Smoothing)

Згладжування ресурсів — це техніка, яка використовується для оптимізації використання ресурсів без зміни загальної тривалості проекту.

Методи згладжування ресурсів:

- Перерозподіл ресурсів між роботами

- Зміна послідовності робіт у межах резерву часу
- Використання альтернативних ресурсів
- Зміна графіка роботи ресурсів

7.5.3. Розподіл ресурсів між проектами

У мультипроектному середовищі ресурси часто повинні розподілятися між кількома проектами. Методи розподілу ресурсів між проектами:

- **Пріоритезація проектів** — розподіл ресурсів на основі відносної важливості проектів
- **Часткове виділення ресурсів** — розподіл часу ресурсу між кількома проектами
- **Послідовне виконання проектів** — планування проектів у послідовності, яка оптимізує використання ресурсів
- **Пул ресурсів** — створення спільного пулу ресурсів, які розподіляються між проектами за необхідності

7.6. Інструменти планування ресурсів

Для планування ресурсів використовуються різноманітні інструменти:

1. **Ресурсні гістограми** — графіки, які показують використання ресурсів у часі
2. **Ресурсні календарі** — календарі, які показують доступність ресурсів у часі
3. **Матриці розподілу ресурсів** — таблиці, які показують розподіл ресурсів за роботами
4. **Програмні засоби управління проектами** — спеціалізоване ПЗ для планування та управління ресурсами
5. **Системи управління ресурсами підприємства (ERP)** — інтегровані системи для управління всіма ресурсами організації

8. Оцінка економічної ефективності проекту

8.1. Значення оцінки економічної ефективності

Оцінка економічної ефективності проекту — це процес аналізу економічних вигод та витрат проекту для визначення його економічної доцільності та порівняння з альтернативними варіантами.

Оцінка економічної ефективності дозволяє:

- Обґрунтувати доцільність інвестицій у проект
- Порівняти різні варіанти реалізації проекту
- Визначити оптимальний обсяг та терміни інвестицій
- Оцінити ризики проекту
- Прийняти обґрунтовані інвестиційні рішення

8.2. Фінансові показники проекту

Основні фінансові показники, які використовуються для оцінки економічної ефективності проекту:

8.2.1. Чиста приведена вартість (NPV)

Чиста приведена вартість (Net Present Value, NPV) — це сума дисконтованих значень потоку платежів, приведених до сьогоднішнього дня.

NPV розраховується за формулою:

NPV = Σ (Bt - Ct) / (1 + r)^t - IC

де:

- Bt — вигоди в період t
- Ct — витрати в період t
- r — ставка дисконтування
- t — період часу
- IC — початкові інвестиції

Проект вважається економічно ефективним, якщо NPV > 0.

8.2.2. Індекс прибутковості (PI)

Індекс прибутковості (Profitability Index, PI) — це відношення приведеної вартості майбутніх грошових потоків до початкових інвестицій.
PI розраховується за формулою:
PI = Σ (Bt - Ct) / (1 + r)^t / IC
Проект вважається економічно ефективним, якщо PI > 1.

8.2.3. Внутрішня норма прибутковості (IRR)

Внутрішня норма прибутковості (Internal Rate of Return, IRR) — це ставка дисконтування, при якій NPV проекту дорівнює нулю.
IRR визначається з рівняння:
Σ (Bt - Ct) / (1 + IRR)^t - IC = 0
Проект вважається економічно ефективним, якщо IRR > r (ставка дисконтування).

8.2.4. Дисконтований період окупності (DPP)

Дисконтований період окупності (Discounted Payback Period, DPP) — це період часу, необхідний для того, щоб дисконтовані вигоди проекту перевищили дисконтовані витрати.
DPP визначається як найменше значення n, для якого виконується умова:
Σ (Bt - Ct) / (1 + r)^t ≥ IC, t = 1 to n
Проект вважається економічно ефективним, якщо DPP < максимально допустимого періоду окупності.

8.3. Методи оцінки економічної ефективності IT-проектів

8.3.1. Традиційні фінансові методи

Традиційні фінансові методи базуються на аналізі грошових потоків та включають розрахунок NPV, IRR, PI та DPP, як описано вище.
Переваги:

- Широке використання та розуміння
- Можливість порівняння різних проектів
- Врахування часової вартості грошей

Недоліки:

- Складність прогнозування грошових потоків
- Складність визначення ставки дисконтування
- Недостатнє врахування нематеріальних вигод

8.3.2. Аналіз вигод та витрат (CBA)

Аналіз вигод та витрат (Cost-Benefit Analysis, CBA) — це метод, який порівнює вигоди та витрати проекту, виражені в грошовій формі.
Етапи CBA:

1. Визначення всіх вигод та витрат
2. Оцінка вигод та витрат у грошовій формі
3. Дисконтування майбутніх вигод та витрат
4. Розрахунок показників ефективності (NPV, BCR, IRR)
5. Аналіз чутливості

BCR (Benefit-Cost Ratio) — відношення дисконтованих вигод до дисконтованих витрат.

8.3.3. Сукупна вартість володіння (TCO)

Сукупна вартість володіння (Total Cost of Ownership, TCO) — це метод, який враховує всі витрати, пов'язані з володінням та використанням системи протягом її життєвого циклу.
Компоненти TCO:

- Початкові інвестиції (придбання, розробка)
- Витрати на впровадження
- Витрати на експлуатацію
- Витрати на підтримку та обслуговування
- Витрати на модернізацію
- Витрати на виведення з експлуатації

8.3.4. Економічна додана вартість (EVA)

Економічна додана вартість (Economic Value Added, EVA) — це метод, який оцінює економічний прибуток проекту з урахуванням вартості капіталу.

EVA розраховується за формулою:

EVA = NOPAT - (WACC * IC)

де:

- NOPAT — чистий операційний прибуток після оподаткування
- WACC — середньозважена вартість капіталу
- IC — інвестований капітал

8.3.5. Соціальна ставка дисконтування (SRD)

Соціальна ставка дисконтування (Social Rate of Discount, SRD) — це метод, який враховує соціальні вигоди та витрати проекту.

SRD враховує:

- Часові уподобання суспільства
- Альтернативні соціальні витрати капіталу
- Соціальну цінність суспільних благ
- Міжпоколінну справедливість

8.4. Особливості оцінки ефективності проектів розробки програмних систем

8.4.1. Врахування нематеріальних вигод

Проекти розробки програмних систем часто мають значні нематеріальні вигоди, які складно виразити в грошовій формі:

- Підвищення якості продукції
- Покращення обслуговування клієнтів
- Підвищення задоволеності користувачів
- Покращення іміджу організації
- Підвищення конкурентоспроможності
- Підвищення гнучкості бізнес-процесів

Для врахування нематеріальних вигод використовуються різні підходи:

- Експертні оцінки
- Методи непрямої оцінки
- Методи умовної оцінки
- Аналіз сценаріїв

8.4.2. Управління вартістю проекту

Управління вартістю проекту включає:

1. **Планування вартості**:
    - Оцінка витрат
    - Розробка бюджету
    - Визначення резервів
2. **Моніторинг та контроль вартості**:
    - Відстеження фактичних витрат
    - Порівняння з плановими витратами
    - Аналіз відхилень
    - Прогнозування кінцевої вартості
3. **Управління змінами вартості**:
    - Оцінка впливу змін на вартість
    - Затвердження або відхилення змін
    - Оновлення базового плану за необхідності

8.4.3. Аналіз ризиків та невизначеності

Аналіз ризиків та невизначеності є важливою частиною оцінки економічної ефективності, особливо для проектів розробки програмних систем, які характеризуються високим рівнем невизначеності.

Методи аналізу ризиків:

1. **Аналіз чутливості** — аналіз впливу зміни окремих параметрів на показники ефективності
2. **Аналіз сценаріїв** — аналіз показників ефективності для різних сценаріїв розвитку подій (оптимістичний, песимістичний, найбільш ймовірний)
3. **Метод Монте-Карло** — статистичне моделювання, яке дозволяє оцінити розподіл ймовірностей показників ефективності
4. **Дерева рішень** — графічний метод аналізу послідовності рішень та їх наслідків

9. Висновки

Планування є одним із найважливіших етапів управління проектом розробки програмних систем. Ефективне планування забезпечує чітке розуміння цілей, обсягу, термінів, ресурсів та бюджету проекту, а також створює основу для моніторингу та контролю виконання проекту.

Основні аспекти планування проекту включають:

- Планування часу, яке визначає терміни виконання робіт та загальну тривалість проекту
- Планування вартості, яке визначає бюджет проекту та розподіл витрат
- Планування якості, яке визначає вимоги до якості продукту та процесів
- Розробку документації, яка забезпечує формалізацію та комунікацію планів та результатів
- Календарне планування, яке визначає конкретні дати початку та завершення робіт
- Планування ресурсів, яке забезпечує наявність необхідних ресурсів у потрібний час
- Оцінку економічної ефективності, яка обґрунтовує доцільність інвестицій у проект

У проектах розробки програмних систем планування має свої особливості, пов'язані з нематеріальною природою програмного забезпечення, високою мінливістю вимог, складністю оцінки трудомісткості та залежністю від людського фактора. Ефективне планування таких проектів вимагає використання специфічних методів та інструментів, а також адаптації традиційних підходів до особливостей розробки програмного забезпечення.

Сучасні підходи до планування проектів розробки програмних систем включають:

- Використання гнучких (Agile) методологій для адаптації до змін
- Інкрементальне та ітеративне планування
- Використання автоматизованих інструментів планування
- Залучення команди до процесу планування
- Безперервне вдосконалення процесів планування на основі отриманого досвіду

Ефективне планування є ключовим фактором успіху проекту розробки програмних систем, оскільки забезпечує основу для координації зусиль усіх учасників проекту, оптимального використання ресурсів та досягнення цілей проекту в межах встановлених обмежень за часом, вартістю та якістю.

10. Список використаних джерел

1. Project Management Institute. (2021). A Guide to the Project Management Body of Knowledge (PMBOK Guide) (7th ed.). Project Management Institute.
2. Highsmith, J. (2009). Agile Project Management: Creating Innovative Products (2nd ed.). Addison-Wesley Professional.
3. Cohn, M. (2005). Agile Estimating and Planning. Prentice Hall.
4. DeMarco, T. (2002). Slack: Getting Past Burnout, Busywork, and the Myth of Total Efficiency. Broadway Books.
5. Goldratt, E. M. (1997). Critical Chain. North River Press.
6. McConnell, S. (2006). Software Estimation: Demystifying the Black Art. Microsoft Press.

7. IEEE Computer Society. (2014). Software Engineering Body of Knowledge (SWEBOK V3.0). IEEE Computer Society.
8. ISO/IEC/IEEE 12207:2017. (2017). Systems and software engineering — Software life cycle processes. International Organization for Standardization.
9. ISO/IEC 25010:2011. (2011). Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. International Organization for Standardization.
10. Boehm, B. W. (1981). Software Engineering Economics. Prentice Hall.
11. Fleming, Q. W., & Koppelman, J. M. (2010). Earned Value Project Management (4th ed.). Project Management Institute.
12. Brooks, F. P. (1995). The Mythical Man-Month: Essays on Software Engineering (Anniversary Edition). Addison-Wesley Professional.