Крижановський Є. М., Ящолт А. Р., Жуков С. О.

Моделювання бізнес-процесів та управління IT-проєктами

Міністерство освіти і науки України
Вінницький національний технічний університет

Крижановський Є. М., Ящолт А. Р., Жуков С. О.

МОДЕЛЮВАННЯ БІЗНЕС-ПРОЦЕСІВ ТА УПРАВЛІННЯ ІТ-ПРОЄКТАМИ

Електронний навчальний посібник комбінованого (локального та мережного) використання

Вінниця
ВНТУ
2022

УДК 338.28:303.447.3(075)
К74

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 11 від 30.06.2022 р.)

Рецензенти:

О. В. Бісікало, доктор технічних наук, професор ВНТУ
Е. М. Деркач, доктор юридичних наук, професор ДОННУ
І. В. Варчук, кандидат технічних наук, доцент, ВНТУ

Крижановський, Є. М.

К74 Моделювання бізнес-процесів та управління ІТ-проєктами : електронний навчальний посібник комбінованого (локального та мережного) використання [Електронний ресурс]. Вид. 2-ге, змін. та доповн. / Є. М. Крижановський, А. Р. Ящолт, С. О. Жуков. – Вінниця : ВНТУ, 2022. – 129 с.

У навчальному посібнику викладено основні теоретичні відомості про методи та засоби моделювання бізнес-процесів та управління ІТ-проєктами, а також практичні рекомендації для застосування сучасних технологій при моделюванні бізнес-процесів та управлінні ІТ-проєктами різної складності. Наведено перелік контрольних запитань для перевірки набутих теоретичних знань та практичних навичок.

Посібник рекомендується для студентів, які навчаються за спеціальностями 126 – «Інформаційні системи та технології» та 124 – «Системний аналіз», при вивченні дисциплін «Організація та управління ІТ-проєктами», «Автоматизація бізнес-процесів», «Управління ІТ-проєктами», «Моделювання бізнес-процесів та управління ІТ-проєктами», «Технології та системи обробки даних», «Створення аналітичних веб-систем», «Технології геоінформаційного аналізу даних».

УДК 338.28:303.447.3(075)

ЗМІСТ

ВСТУП

Моделювання бізнес-процесів та управління ІТ-проєктами є одними з найпоширеніших напрямків застосування сучасних технічних засобів в галузі інформаційних технологій.

Моделювання бізнес-процесів в умовах сучасності неможливі без використання комп'ютерної техніки та сучасних програмних засобів. Сучасні пакети прикладних програм дозволяють здійснювати обробку даних з використанням традиційних і сучасних математичних методів для моделювання бізнес-процесів. Застосування сучасних інформаційних технологій також є необхідним для здійснення ефективного системного управління ІТ-проєктами.

Посібник орієнтовано на студентів комп'ютерних спеціальностей, які повинні вміти використовувати сучасні програмні середовища для моделювання бізнес-процесів різної складності, а також для здійснення управління ІТ-проєктами. Вашій увазі пропонується друге видання, яке базується на основі першого [1]. У другій версії посібника було змінено структуру та наповнення 2-го розділу, переформатовано структуру першого та третього розділів, покращено ілюстративні матеріали.

Колектив авторів висловлює подяку Козачку Олексію Миколайовичу за участь і підготовку матеріалів першої версії посібника та їх використання у поточній версії.

Кафедра системного аналізу та інформаційних технологій Факультету інтелектуальних інформаційних технологій та автоматизації ВНТУ має великий досвід застосування усього викладеного матеріалу в даному посібнику на різних проєктах у різних сферах (більш детально про проєкти кафедри можна знайти за посиланням https://mmss.vntu.edu.ua/index.php/ua/proekti-kafedri).

Автори посібника використали досвід, який набули під час стажування в Університеті Тарту (Естонія) у межах проєкту «Development of innovative capacity and entrepreneurial competence of Ukrainian universities: sharing best practice of Estonia (UnivEntre)».

Мета даного посібника – ознайомити студентів із основними знаннями та навичками, необхідними для здійснення моделювання бізнес-процесів різної складності, а також для здійснення управління ІТ-проєктами.

Матеріал посібника може бути корисним і для слухачів другої вищої освіти інформаційних систем та технологій, чи системного аналізу та

студентам інших спеціальностей, які недостатньо володіють основними знаннями та навичками роботи з найбільш поширеними в Україні програмними засобами, що можуть бути використані для проведення моделювання бізнес-процесів й управління ІТ-проєктами. Також буде корисним слухачам таких курсів: «Організація та управління ІТ-проєктами», «Автоматизація бізнес-процесів», «Управління ІТ-проєктами», «Моделювання бізнес-процесів та управління ІТ-проєктами», «Технології та системи обробки даних», «Створення аналітичних веб-систем», «Технології геоінформаційного аналізу даних».

1 ПРИНЦИПИ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ

1.1 Основні положення управління проєктами

Проєкти часто використовуються як засіб прямого або непрямого, або декількох з нижчевказаних стратегічних міркувань [1]:

– вимога ринку (наприклад, автомобілебудівна компанія авторизує проєкт з виготовлення більш економічних автомобілів у відповідь на брак бензину);

– стратегічна можливість / бізнес-потреба (наприклад, тренінгова компанія авторизує проєкт зі створення нового курсу навчання з метою збільшення доходів);

– соціальна потреба (наприклад, неурядова організація в країні, що розвивається, авторизує проєкт з надання систем питного водопостачання, туалетів і санітарної освіти співтовариствам, що страждають від високого рівня інфекційних захворювань);

– захист навколишнього середовища (наприклад, державна компанія авторизує проєкт зі створення нового сервісного центру для електромобілів, які сприяють скороченню забруднення навколишнього середовища);

– вимога замовника (наприклад, компанія-виробник електроенергії для громадського користування авторизує проєкт з будівництва нової підстанції для електропостачання нового промислового району);

– технологічний прогрес (наприклад, виробник комп'ютерної техніки авторизує проєкт з розробки більш швидкодійного, економічного і компактного ноутбука з використанням досягнень в технології виготовлення комп'ютерної пам'яті та електронних компонентів);

– юридична вимога (наприклад, виробник хімічних речовин авторизує проєкт з розробки керівних вказівок щодо поводження з новим токсичним матеріалом).

Організації здійснюють керівництво для визначення стратегічного напрямку і параметрів продуктивності. Даний стратегічний напрям надає мету, очікування, завдання і дії, необхідні для керівництва діяльністю організації, і приводиться у відповідність з бізнес-цілями. Роботи з управління проєктом мають бути приведені у відповідність з напрямком організації на верхньому рівні, і в разі його зміни мета і цілі проєкту мають бути переглянуті. В умовах виконання проєкту зміни мети і цілей проєкту впливають на ефективність і успіх проєкту.

Основними принципами нової концепції управління є [1]:

1) поглиблення рівня обґрунтованості прийнятих інвестиційних рішень, використовуючи механізм різноманітних і багатофакторних (технологічних, економічних, соціальних, екологічних та інших) оцінок;

2) високий ступінь координації та контролю робіт в процесі виконання проєкту;

3) систематичний аналіз і врахування зовнішніх змін (кон'юнктури ринку щодо всіх видів ресурсів, непередбачених обставин і негативних факторів) при реалізації проєктів.

Управління проєктами базується на системному підході, що дає можливість декомпозиції і структуризації проєкту будь-якої складності при прийнятті рішень в складних умовах (ситуаціях). Особливість методології управління проєктами полягає в зосередженні прав і відповідальності за досягнення цілей проєкту на одній людині або невеликій групі. Ці права і обов'язки здійснює керівник проєкту. При цьому забезпечується [1]:

1. Визначення всіх видів робіт, необхідних для досягнення цілей проєкту, їх структуризація і визначення взаємозв'язків;

2. Складання і контроль кошторису витрат з реалізації проєкту;

3. Розробка та контроль графіків робіт, необхідних для досягнення бажаного результату;

4. Розподіл ресурсів, виділених для реалізації проєкту, в умовах невизначеностей і ризиків;

5. Управління якістю виконання всіх робіт проєкту, зокрема досягнення запланованої якості продукції, пропонованої проєктом, і виконання інших вимог замовника;

6. Управління ризиком (ризиками) на всіх етапах здійснення проєкту;

7. Забезпечення зв'язку з клієнтами, замовниками, споживачами продукції, з різними групами і особами, залученими до проєкту (соціальні групи, місцеве населення, влади, засоби масової інформації) для вирішення всіх питань, пов'язаних з досягненням успіху проєкту.

При розгляді та вивченні діяльності з управління реалізацією проєктів можна виділити низку підходів, що визначають структуризацію цієї діяльності. Найбільш поширені з них:

1. Функціональний підхід відображає загальний підхід до проблеми управління і передбачає розгляд основних управлінських функцій або видів управлінської діяльності при здійсненні проєктів відповідно до класифікації, прийнятої в менеджменті. У той же час, специфіка управління проєктами проявляється в певній деталізації окремих функцій менеджменту. З урахуванням цього в рамках функціонального підходу розглядаються управлінські дії, які визначаються як функції управління проєктами, мається на увазі: аналіз, планування, організація, контроль і регулювання [1].

2. Предметний підхід визначає структуризацію управлінського процесу на об'єктах проєкту, на які направлено управління. В рамках предметного підходу розглядаються два типи об'єктів: виробничі об'єкти (перший тип) і елементи, пов'язані з діяльністю щодо забезпечення реалізації проєкту (другий тип). Об'єктами другого типу можуть бути: фінанси, кадри, маркетинг, контроль, ризики, матеріальні ресурси, якість, інформація та інші елементи, що забезпечують отримання бажаного результату при

реалізації проєкту. Основні об'єкти першого і другого типів розглядаються, за необхідності, при вивченні дисципліни.

3. Динамічний підхід (відомий в літературі як проєктний менеджмент – project management) передбачає розгляд у часі процесів, пов'язаних з реалізацією основних стадій і етапів реалізації проєкту: аналіз проблеми, розробка концепції проєкту, проєктування, будівництво, монтаж, налагоджувальні роботи, експлуатація та завершення проєкту. Деталізація розглянутих процесів стосовно конкретного проєкту (етапів і стадій робіт здійснення проєкту) може змінюватися в широких межах [1].

1.2 Класифікація ІТ-проєктів, їх особливості

Для зручності аналізу і синтезу проєктів, а також систем управління ними безліч різноманітних ІТ-проєктів класифікується за різними обставинами. У науковій літературі зустрічаються різні підходи до класифікації проєктів. Найбільш часто вживаною класифікацією є [2–12]:

1) Клас. За складом і структурою проєкта та його предметною областю проєкти поділяються на монопроєкти, мультипроєкти, мегапроєкти.

2) Тип. За основними сферами діяльності, в яких здійснюється проєкт, виділяють технічні, організаційні, економічні, соціальні та змішані проєкти.

3) Вид. За характером предметної галузі проєкти поділяються на: інвестиційні, науково-дослідні, навчально-освітні, змішані.

4) Масштаб. За розмірами самого проєкту, кількістю учасників і ступеня впливу на навколишній світ проєкти поділяють на: дрібні, середні, великі й дуже великі проєкти.

5) Тривалість. За тривалістю періоду виконання проєкти поділяються на: короткострокові, середньострокові і довгострокові.

6) Складність. За ступенем складності: прості, складні і дуже складні.

Використання зазначеної класифікації не повною мірою дає уявлення про відмінності ІТ-проєктів від інших видів проєктів. Як наслідок, виникає необхідність індивідуалізації наведеної класифікаційної системи.

ІТ-проєкти можна розділити на три основні типи [2, 6]:

1) розробка програмного забезпечення, комп'ютерних програм, процедур і, можливо, відповідної документації та даних, які відносять до функціонування інформаційної системи;

2) розробка продуктів, орієнтованих на онлайн використання;

3) розробка різних додатків, частіше яскраво вираженого комерційного спрямування.

Усі три види мають свою специфіку. Ґрунтуючись на такий поділ, можна виділити основні класифікаційні ознаки ІТ-проєктів [2–12]:

1) За характером змін: прості (проєкти, здійснення змін у структурі та змісті яких не призводить до зміни їх вартісних і часових параметрів); середні (проєкти, здійснення змін у структурі та змісті яких, призводить до

несуттєвої зміни їх вартісних і часових параметрів); складні (проєкти, здійснення змін у структурі та змісті яких призводить до істотної зміни їх вартісних і часових параметрів).

2) За масштабом: малі (вартістю до 10 тис. грн); середні (вартістю 10–50 тис. грн); великі (вартістю 50–100 тис. грн); значні (вартістю 100–1000 тис. грн); дуже значні (вартістю понад 1 млн грн).

3) За тривалістю: короткострокові – тривалістю до 1-го року; середньострокові – тривалістю від 1-го до 3-х років; довгострокові – тривалістю понад 3 роки.

4) За стадіями життєвого циклу системи: оформлення задуму і концепції; формулювання вимог до системи; розробка системи; введення системи в експлуатацію; підтримка існуючої системи.

5) За видом продукту: система; програмний продукт; технічні засоби; програмно-технічні комплекси; матеріали, роботи та послуги.

6) За функціональним призначенням (орієнтованість продуктів ІТ-проєктів на напрямки діяльності замовника): виробничі; технологічні; фінансові; дослідні; маркетингові; з управління персоналом; з управління проєктами; ігрові; комбіновані.

7) За глибиною взаємного проникнення бізнесу замовника і підрядника: аутсорсинг; рішення «під ключ»; спільні проєкти; сервісна модель; аудит і консалтинг.

8) За видом замовника: держсектор; медицина; освіта; дрібний бізнес; великий бізнес; логістика, сфера послуг; роздріб; енергетика; банківський сектор; транспорт і зв’язок; промисловість; оборонна промисловість тощо.

9) За видом автоматизованих процесів: основні і допоміжні; технологічні та офісні; управлінські; аналітичні; транзакційні; реального часу; з тим чи іншим акцентом на обчислювальну обробку; передача даних; організація зберігання; обробка медіа-контенту; забезпечення безпеки тощо.

10) За ступенем складності: монопроєкти, мультипроєкти і мегапроєкти з притаманними їм загальноприйнятими характеристиками.

11) За територіальним поширенням: мононаціональні – продукт проєкту орієнтований на територіальні регіони зі схожою ментальністю; полінаціональні – продукт проєкту орієнтований на територіальні регіони з різною ментальністю.

12) За рівнем впливу розробки інтерфейсу на проєкт: низький – результат розробки інтерфейсу має незначний вплив на оцінку проєкту в цілому, займає невеликий відрізок часу в життєвому циклі проєкту; середній – результат розробки інтерфейсу має значний вплив на оцінку проєкту в цілому, займає значний відрізок часу в життєвому циклі проєкту; високий – результат розробки інтерфейсу має критично важливий вплив на оцінку проєкту в цілому, займає значний відрізок часу в життєвому циклі проєкту, може вплинути на прийняття рішення про закриття проєкту.

Управління проєктами у сфері інформаційних технологій за останній час завоювало визнання як найкращий метод планування та управління реалізацією інвестиційних проєктів. За американськими оцінками застосування методології управління проєктами забезпечує високу надійність досягнення цілей проєкту і на 10–15% скорочує витрати на його реалізацію [1].

У світі накопичений величезний досвід застосування управління проєктами в галузі інформаційних технологій. Зокрема, ця методологія застосовується в усіх великих IT-компаніях світу. Програмні засоби для управління проєктами встановлені на мільйонах комп'ютерів в усьому світі – тільки пакет Microsoft Project встановлений більш ніж на двох мільйонах комп'ютерів. Асоціація управління проєктами Project Management Institute (Інститут Управління Проєктами) об'єднує близько 40 тисяч членів і має відділення майже на всіх континентах [1, 11].

Розглянемо основні поняття і методи управління проєктами.

Проєкт – це тимчасове підприємство, призначене для створення унікальних продуктів або послуг.

В даному контексті «тимчасове» означає, що у кожного проєкту є початок і неодмінно настає завершення, коли досягаються поставлені цілі, або приходить розуміння, що ці цілі не можуть бути досягнуті.

В даному контексті «унікальних» означає, що створювані продукти або послуги відрізняються від інших аналогічних продуктів і послуг. Приклади проєктів: розробка нового обладнання, розробка або впровадження програмних засобів тощо.

Управління проєктами – це застосування знань, досвіду, методів і засобів до робіт проєкту для задоволення вимог, що висуваються до проєкту, і очікувань учасників проєкту. Щоб задовольнити ці вимоги і очікування потрібно знайти оптимальне поєднання між усіма характеристиками проєкту [11].

Управління проєктами підпорядковується чіткій логікі, яка пов'язує між собою різні галузі знань і процеси управління проєктами.

Кожен проєкт приводить до створення унікального продукту, послуги або результату.

Перш за все, у проєкті обов'язково є одна або кілька цілей. Під цілями розуміються не тільки кінцеві результати проєкту, а й обрані шляхи досягнення цих результатів (наприклад, технології, які застосовувались в проєкті, системи управління проєктом тощо) [1].

Досягнення цілей проєкту може бути реалізовано різними способами. Для порівняння цих способів потрібні критерії успішності досягнення поставлених цілей. Зазвичай в число основних критеріїв оцінювання різних варіантів виконання проєкту входять терміни і вартість досягнення результатів. При цьому заплановані цілі і якість зазвичай служать основними обмеженнями при розгляді та оцінюванні різних варіантів. Звичайно, можливе використання і інших критеріїв, і обмежень, зокрема, ресурсних [1].

Застосування технологій і ресурси проєкту можна віднести до основних важелів управління проєктами. Крім цих основних важелів існують і допоміжні засоби, призначені для управління основними. До таких допоміжних важелів управління можна віднести, наприклад, контракти, які дозволяють залучити потрібні ресурси в потрібні терміни. Крім того, для управління ресурсами необхідно забезпечити ефективну організацію робіт. Це стосується структури управління проєктом, організації інформаційної взаємодії учасників проєкту, управління персоналом.

Інформація, яка використовується в управлінні проєктами, зазвичай не буває стовідсотково достовірною. Врахування невизначеності вихідної інформації потрібне і при плануванні проєкту і для грамотного укладання контрактів. Аналізу та врахуванню невизначеностей присвячений аналіз ризиків [1, 11].

Управління проєктом здійснюється за допомогою логічно згрупованих процесів управління проєктом, об'єднаних в 5 груп [1]:

– Процеси ініціації – ухвалення рішення про початок виконання проєкту;

– Процеси планування – визначення цілей і критеріїв успіху проєкту і розробка робочих схем їх досягнення;

– Процеси виконання – координація людей і інших ресурсів для виконання плану;

– Процеси моніторингу і контролю – визначення відповідності плану і виконання проєкту поставленим цілям і критеріям успіху та прийняття рішень про необхідність застосування коригувальних дій і визначення необхідних коригувальних впливів, їхнє узгодження, затвердження і застосування;

– Процеси закриття – формалізація виконання проєкту і підведення його до впорядкованого фіналу.

Практично методологія управління проєктами допомагає [1]:

– обґрунтувати доцільність інвестицій;

– розробити оптимальну схему фінансування робіт;

– скласти план робіт, що охоплює терміни виконання робіт, споживання ресурсів, необхідні витрати;

– оптимально організувати виконання робіт і взаємодію учасників проєкту;

– здійснювати планування і управління якістю;

– здійснювати аналіз і управління проєктними ризиками;

– оптимально планувати і управляти контрактами;

– аналізувати відхилення фактичного ходу виконання робіт від запланованого і прогнозувати наслідки виникнення відхилень;

– моделювати коригувальні дії на інформаційних моделях проєктів і приймати обґрунтовані управлінські рішення;

– вести архіви проєктів і аналізувати досвід їх реалізації, який може бути використаний в інших проєктах тощо.

1.3 Життєвий цикл проєкту

Життєвий цикл проєкту – набір фаз, через які проходить проєкт з моменту його ініціації до моменту закриття. Фази, як правило, є послідовними, а їх назви та кількість визначаються потребами в управлінні і контролі організації або організацій, залучених до проєкту, характером самого проєкту і його прикладною сферою. Проєкт може бути розбитий на фази залежно від функціональних або часткових цілей, проміжних результатів, визначених контрольних подій всередині загального змісту робіт або доступності фінансів. Фази, як правило, обмежені в часі, і мають початкову і кінцеву або контрольну точки. Життєвий цикл може документуватися в рамках методології. Життєвий цикл проєкту може визначатися або формуватися унікальними аспектами організації, галузі або використовуваної технології. У той час як кожен проєкт має певний початок і закінчення, конкретні результати і дії, що мають місце в цьому проміжку, широко варіюються для кожного проєкту. Життєвий цикл забезпечує базову структуру для управління проєктом, незалежно від внесених до нього конкретних робіт [1].

Проєкти розрізняються за розміром і складністю. Всі проєкти можуть мати таку структуру життєвого циклу (рис. 1.1):

– початок проєкту;

– організація та підготовка;

– виконання робіт проєкту;

– завершення проєкту.

Дана узагальнена структура життєвого циклу часто згадується при комунікаціях з вищим керівництвом або іншими сторонами, які менш обізнані про деталі проєкту. Не потрібно плутати її з групами процесів управління проєктом, тому що процеси в групі процесів складаються з дій, які можуть виконуватися і повторюватися в кожній фазі проєкту, а також бути характерними для проєкту в цілому. Життєвий цикл проєкту не залежить від життєвого циклу продукту, виробленого або модифікованого в результаті виконання проєкту. Однак проєкт має враховувати поточну фазу життєвого циклу продукту. Це високорівневе подання забезпечує єдину систему відліку при порівнянні проєктів, навіть якщо вони різнорідні за своєю природою [1].

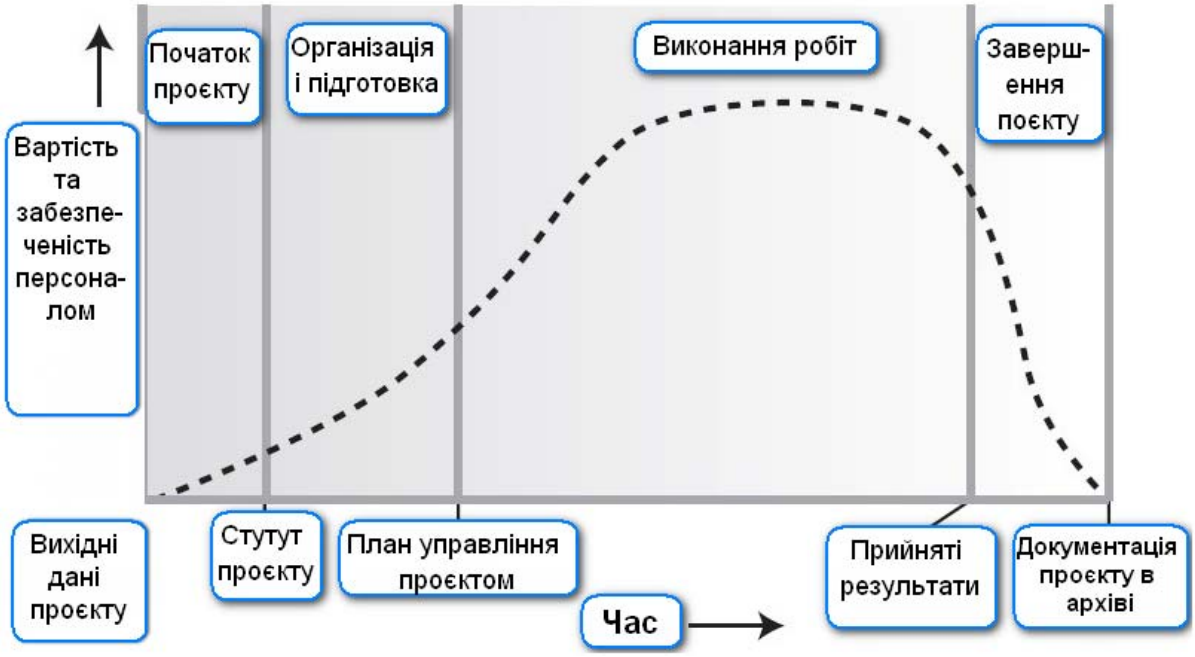


Рисунок 1.1 – Життєвий цикл проєкту

На етапі початку проєкту важливо використовувати сучасні підходи до аналізу та пропрацювання власної ідеї. Досить зручно це робити за допомогою таких шаблонів: цінність пропозиції (рис. 1.2), профіль користувача (рис. 1.3), подання бізнес-моделі проєкту (рис. 1.4) та інші [13].

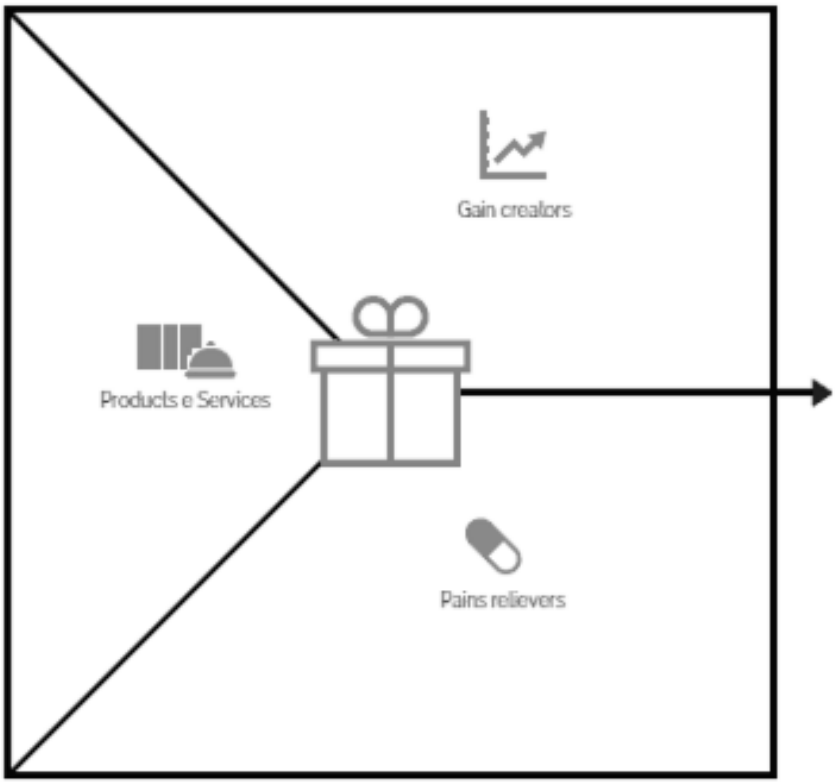


Рисунок 1.2 – Шаблон пропрацювання цінності пропозиції

Рисунок 1.3 – Шаблон пропрацювання профілю користувача

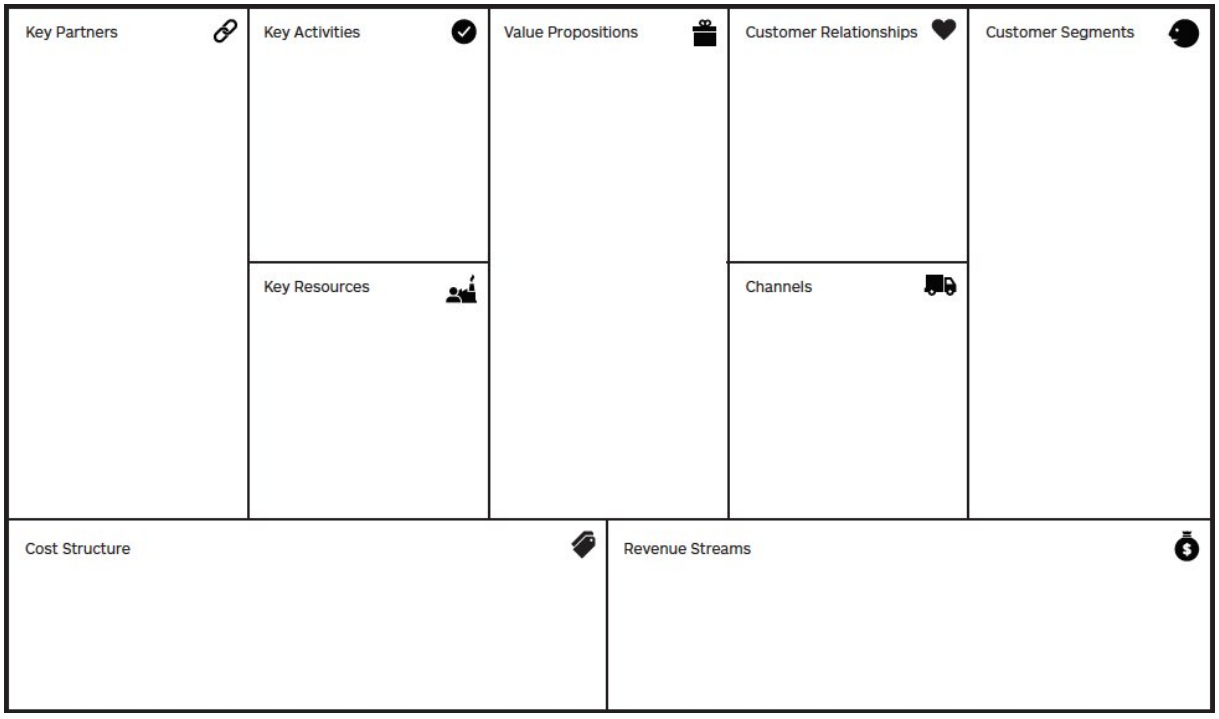


Рисунок 1.4 – Шаблон бізнес-моделі проєкту

Нині для підприємств характерні такі проблеми управління IT-проєктами [14]:

– кількість проєктів, керованих IT-підрозділами, завелика, немає достатньої кількості кваліфікованих менеджерів;

– наявна методологія не забезпечує відповідності ІТ-проєктів цілям компанії, не створена основа для оцінювання повернення інвестицій, вкладених в ІТ;

– роботи виконуються за різними технологіями, рівень керованості недостатній;

– відсутня оперативна аналітична звітність і накопичення досвіду за проєктами.

Серед проблем, суттєвих з погляду успіху проєкту в цілому, можна виділити такі:

– прагнення керівництва підприємства досягнення бізнес-цілей, поставлених перед ІТ-проєктом;

– відповідність функціональності інформаційної системи (ІС) потребам бізнесу і керівництва підприємства;

– управління в межах проєкту (терміни, бюджет, склад робіт), тому що ІТ-проєкти мають властивість «розповзатися».

Це відбувається з різних причин, серед яких можуть бути:

– зміна організаційної структури і реінжиніринг бізнес-процесів підприємства, відсутність деталізованого проєкту ІС, неконтрольований потік вимог у процесі виконання проєкту з боку замовника, зміна проєктних рішень розробниками системи в ході її реалізації, відсутність типізації проєктних рішень;

– взаємодія різних груп учасників ІТ-проєкту. У великому ІТ-проєкті може брати участь кілька сотень і тисяч осіб. Умовно їх можна розділити на декілька груп: розробники ІС, аналітики і методологи, функціональні фахівці замовника, фахівці з навчання і супроводу ІС. Кожна група є носієм певної категорії знань і виконує певні ролі у проєкті. Внаслідок різного професійного досвіду, термінологічного базису, покладених функціональних завдань, у спілкуванні між представниками різних груп часто викликають проблеми, що може призводити до конфліктних ситуацій;

– об'єктивний моніторинг поточного стану проєкту. Часто керівництво ІТ-проєкту не має ефективних засобів доступу до адекватної інформації про поточний стан проєкту, що перешкоджає формуванню своєчасних управлінських рішень щодо проєкту;

– ведення проєктної документації. У багатьох ІТ-проєктах висуваються невисокі вимоги до ведення проєктної документації. Це негативно позначається на етапах супроводу і розвитку ІС, на навчанні кінцевих користувачів, призводить до зміни учасників проєктної команди;

– склад і кваліфікація учасників проєкту, функціональні обмеження типових рішень ІС тощо [14].

1.4 Учасники проєкту

Головним відповідальним і головною дійовою особою в проєкті, через мізки якого мають пройти всі елементи проєкту, є керівник (менеджер)

проєкту. Він призначається на першій фазі (див. рис. 1.1) одночасно з затвердженням статуту проєкту [1].

Роль керівника проєкту – особи, відповідальної за успіх всього проєкту, – набагато ширша ролі функціонального менеджера. Різниця визначається тим, що менеджер проєкту [1]:

– управляє тимчасовою діяльністю і командою тимчасових учасників та фахівців з різних галузей знань;

– має заздалегідь поставлену мету;

– обмежений термінами, бюджетом і технічними умовами (вимогами до продукту і його якості);

– управляє інтеграцією всіх елементів проєкту, сам планує роботу і використання ресурсів і сам втілює їх у життя;

– керує створенням нового унікального продукту, властивості якого можуть уточнюватись в ході виконання проєкту;

– не завжди розуміється в тонкощах реалізації продукту.

Для ефективного виконання своєї ролі менеджер проєкту має мати знання та реальний досвід в таких сферах:

– проєктний менеджмент – технологічні знання і методи управління проєктами;

– загальний менеджмент – знання і методи повсякденного управління компаніями;

– конкретні прикладні, предметні галузі.

Проєктний менеджмент (управління проєктами) – організаційна діяльність з оперативного управління проєктом в умовах обмежень.

Сьогодні проєктний менеджмент багато в чому спирається на стандарти PMBOK (Project Management Body Of Knowledge). Практичне знання стандартів і наявність досвіду та знань з управління проєктами дає менеджеру проєкту значну перевагу перед іншими менеджерами, зокрема при працевлаштуванні [14–15].

Загальний менеджмент охоплює всі аспекти управління повсякденною діяльністю підприємства. Приблизний перелік знань і навичок, що входять в програми MBA (Master of Business Administration) – майстер ділового адміністрування, демонструє ємність завдань загального менеджменту [1]:

– планування часу і делегування повноважень;

– особиста ефективність, проведення нарад і особиста мотивація;

– подолання проблем і прийняття рішень;

– мистецтво ведення переговорів, ефективного спілкування, публічних виступів, листування;

– фінансовий аналіз і бухгалтерський облік, оцінювання інвестицій;

– бюджетування, бюджетний контроль;

– управління продажами і маркетинг;

– управління запасами і незавершеним виробництвом;

– дослідження і розробки, фінансовий аналіз проєктів;

– фінансування, управління прибутком;

– управління персоналом, організаційне управління;

– конкурентна ринкова стратегія, оцінювання бізнесу, покупка і продаж компаній;

– стратегічне планування бізнесу, розробка бізнес-планів;

– методики управління підприємством, зокрема методика BSC (Balanced Score Card) – СЗП (Система збалансованих показників);

– управління змінами, кризовий менеджмент тощо.

Знання та навички загального менеджменту складають хорошу основу для оволодіння знаннями і навичками управління проєктами. Проєктний менеджмент розвивається і розглядається як окрема галузь знань, навичок, компетенцій, що стоїть поруч і на службі таких традиційних сфер професійної діяльності, як будівництво і архітектура, IT-індустрія, соціологія тощо [1].

Крім менеджера проєкту, в проєкті зазвичай беруть участь безліч інших осіб і організацій. До учасників проєкту належать фізичні та юридичні особи, залучені до проєкту, а також особи, які мають вплив (позитивний чи негативний) на проєкт і його результати. Їх в термінології PMI (Project Management Institute) PMBOK називають стейкхолдерами проєкту [15].

Виявити всіх стейкхолдерів проєкту та їх інтереси часто буває важко, і це є однією з перших завдань менеджера проєкту. До ключових стейкхолдерів належать (рис. 1.5) [1]:

– менеджер проєкту – особа, відповідальна за кінцеві результати проєкту та управляє проєктом;

– замовник – фізична або юридична особа – майбутній споживач продукту проєкту;

– підрядник – юридична особа, співробітники якої виконують роботи проєкту; організацією, що виконує роботи, може виступати як зовнішня організація, так і тимчасова структура всередині самої організації-замовника;

– спонсор – особа або група осіб (фізичних або юридичних), що забезпечує проєкт фінансовими та іншими ресурсами;

– члени команди проєкту – група, яка виконує роботи проєкту.

До завдань менеджера проєкту входить і управління стейкхолдерами. Воно розглядається як профілактичне завдання, спрямоване на максимальне врахування інтересів стейкхолдерів, використання їх активності для досягнення цілей проєкту, нейтралізації їх негативного впливу. Воно складається як мінімум з таких пунктів [1]:

– визначення стейкхолдерів, а також оцінення їх компетентності, знань і навичок;

– аналіз проєкту на відповідність вимогам стейкхолдерів;

– залучення стейкхолдерів в проєкт різними шляхами: як експертів, як членів комісій щодо змін тощо;

– якщо існують розбіжності між стейкхолдерами, то за допомогою компромісів проблема має вирішуватися на користь замовника.

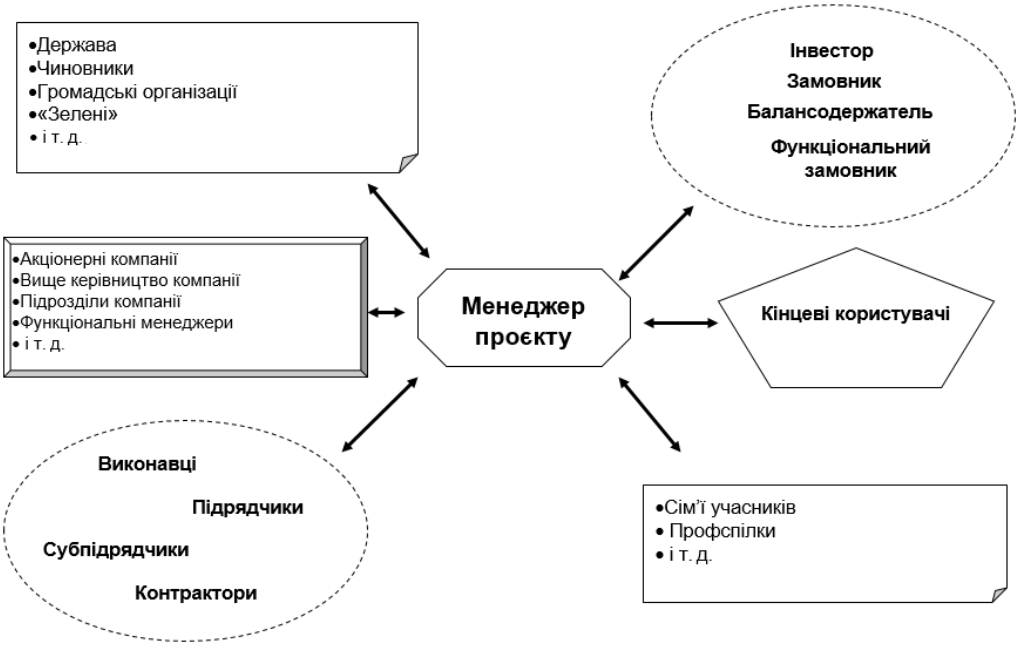


Рисунок 1.5 – Приклад зацікавлених сторін

Команда проєкту (project team) – специфічна організаційна структура (рис. 1.6), сукупність окремих осіб, груп і/або організацій, залучених до виконання робіт проєкту і відповідальних перед керівником проєкту за їх виконання. Створюється цільовим чином на період здійснення проєкту. Охоплює також всіх зовнішніх виконавців і консультантів.

Рисунок 1.6 – Структура команди проєкту

Важливо розуміти, що при реалізації проєкту кожен учасник управляє своїм проєктом. Візуально це можна подати так (рис. 1.7). Інвестор реалізує інвестиційний проєкт, замовник – проєкт зі створення та здачі в експлуатацію продукту, підрядник – проєкт з виконання робіт, а Балансоутримувач – веде проєктний облік і експлуатує створений продукт.

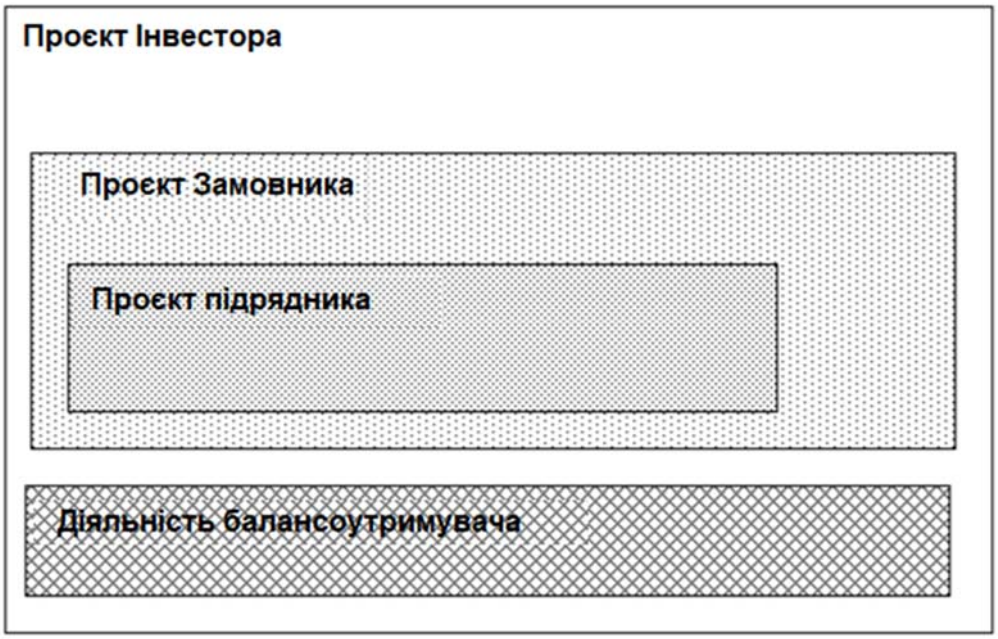


Рисунок 1.7 – Структура команди проєкту

1.5 Форми організаційної структури

Структура організації-виконавця накладає обмеження на процеси управління проєктом і розподіл ресурсів. Позначимо дві з них – функціональна і проєктна. На рисунках 1.8, 1.9 сірим кольором виділено персонал, який бере участь в проєктах. У функціональній структурі (рис. 1.8) кожен такий працівник має одного керівника і взаємодіє тільки з ним. Зазвичай функціональний підрозділ виконує частину робіт проєкту або весь проєкт. Функціональний керівник вирішує всі питання розподілу ресурсів, завдань, взаємодії і залучення фахівців з інших функціональних підрозділів. Роль менеджера проєкту розмита. У таких випадках роль координатора проєкту покладається на експедитора. У той же час експедитор не може приймати самостійні рішення щодо проєкту. Якщо ж призначається менеджер проєкту, то він повинен мати певні повноваження управління командою, максимально чітко встановлені [1].

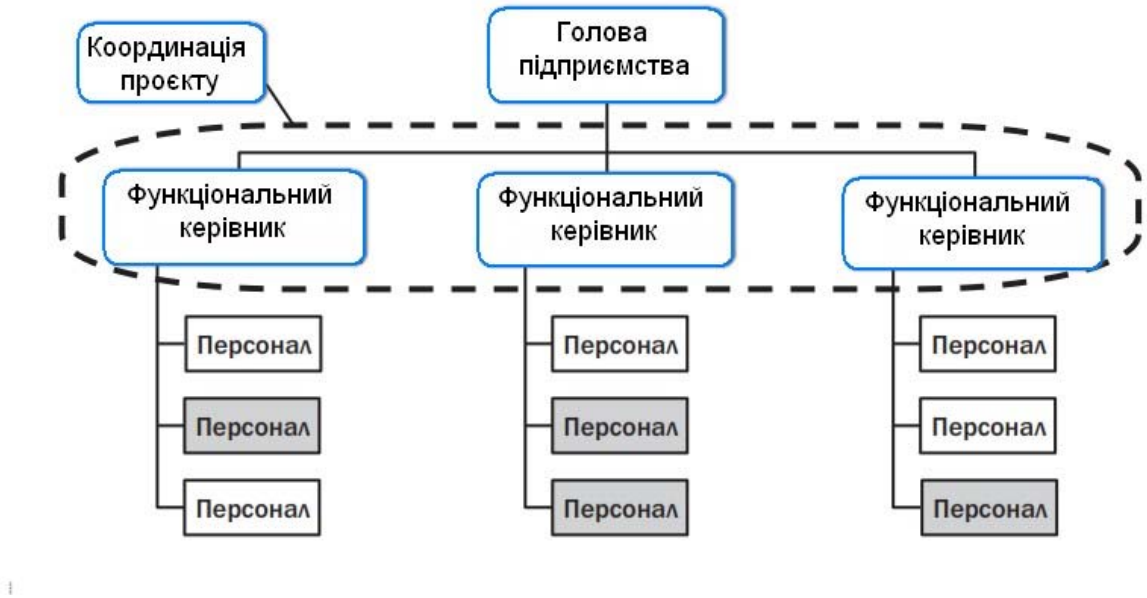


Рисунок 1.8 – Функціональна структура

У проєктній структурі (рис. 1.9) весь персонал на проєкт зібраний в одному місці (команда проєкту). Створені всі умови для тісної взаємодії персоналу різних спеціальностей. Менеджер проєкту достатньо незалежний і наділений максимальними повноваженнями. У багатьох випадках таке формування реалізується у вигляді проєктного офісу.

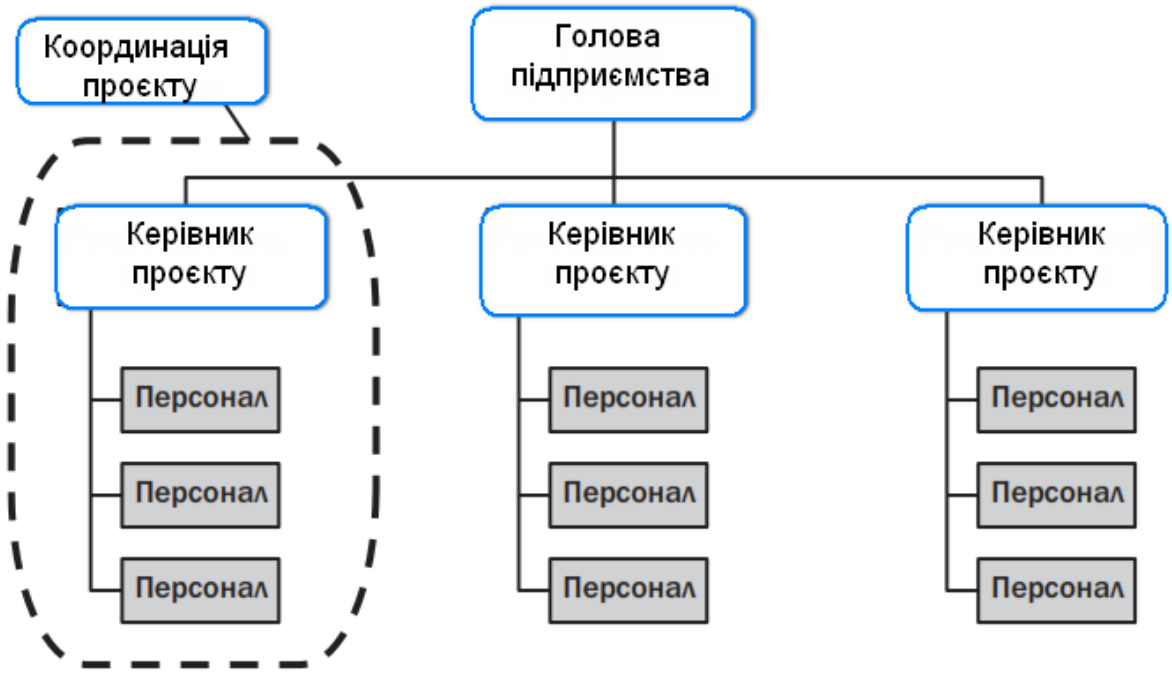


Рисунок 1.9 – Проєктна структура

Функціональна структура більше орієнтована на виконання повторюваних циклічних робіт. Переваги функціональної структури стають недоліками проєктної та навпаки (табл. 1.1).

Таблиця 1.1 – Порівняння організаційних структур

Проєктна організаційна структура	Функціональна організаційна структура
Ефективна інтеграція, організація і контроль проекту	Складніший моніторинг і контроль проєкту. Акцент на функціональну спеціалізацію зі шкодою для інтеграції та інших робіт проєкту
Ефективна комунікація та рішення конфліктів	Неефективні комунікації на ієрархічній драбині
Націленість на проєкт і лояльність проєкту	Розмитість пріоритетів виконання робіт
Імовірність браку професіоналізму з різних дисциплін	Високий професіоналізм, але акцент на функціональну спеціалізацію зі шкодою для інтеграції та інших робіт проєкту
Імовірність надлишку і менш ефективного використання ресурсів	Повна завантаженість фахівців і більш легке управління ними
Тимчасова робота	Наявність постійної роботи після завершення проєкту
Одна відповідальна особа – менеджер проєкту	Розмита відповідальність за результати проєкту

Тому на практиці часто застосовують комбінацію зазначених вище структур – комбінована організаційна структура, або інші поєднання – слабку матричну структуру або сильну матричну структуру. Комбінована структура дозволяє мінімізувати недоліки описаних структур показана на рисунку 1.10.

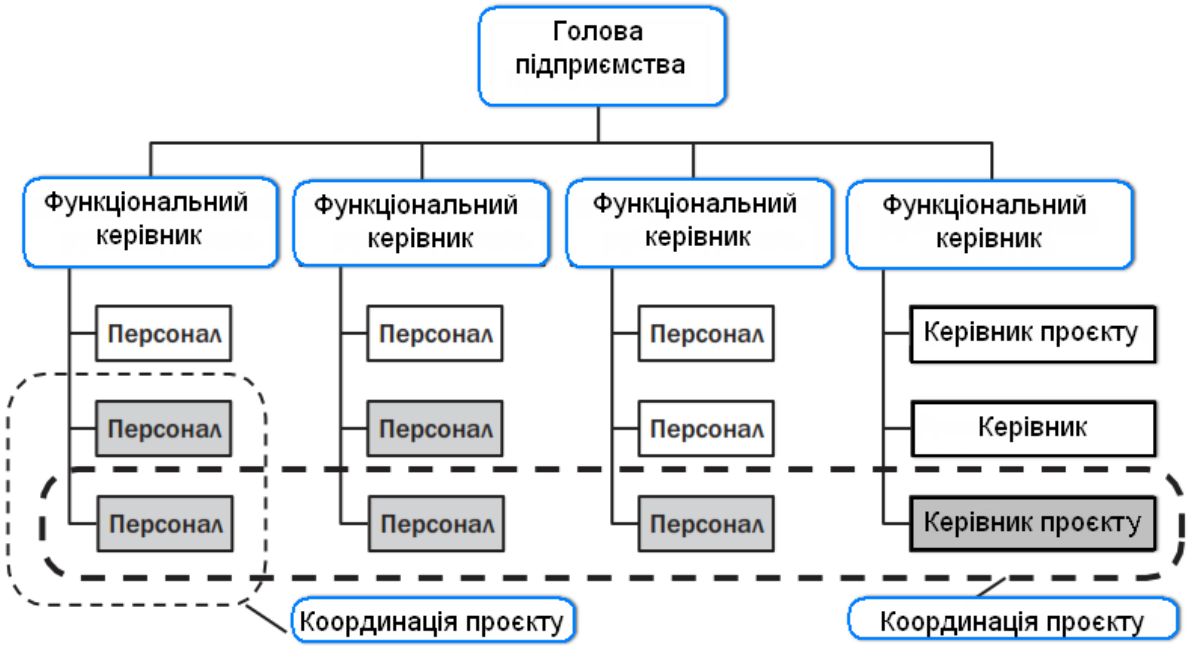


Рисунок 1.10 – Комбінована організаційна структура

У комбінованій структурі персонал на проєкті може мати подвійне підпорядкування: в рамках організації підпорядкування функціональному керівнику (менеджеру), а в рамках проєкту – керівнику (менеджеру) проєкту. Тому мають бути чітко розмежовані повноваження між керівником проєкту і функціональними керівниками. Керівник (див. рис. 1.10) проєкту чітко планує і погоджує участь персоналу в проєкті з функціональними керівниками [1].

Якщо рівень організаційної культури організації не дозволяє розмежувати повноваження менеджерів проєкту і функціональних менеджерів, не допускає нововведень для досягнення успіху, то кращим рішенням є створення незалежних проєктних команд для виконання критично важливих проєктів.

1.6 Стандарти управління проєктами

Методологія управління проєктами відбивається в стандартах управління проєктами. В даний час існують такі види стандартів [1]:

– міжнародні – стандарти, які набули міжнародного значення в процесі свого розвитку або призначені для міжнародного використання;

– національні – створені для застосування всередині однієї країни або отримали загальнонаціональний статус в процесі свого розвитку;

– громадські – підготовлені і прийняті спільнотою фахівців;

– приватні – комплекси знань, пропаговані для вільного використання приватними особами, компаніями або установами;

– корпоративні – розроблені для застосування в межах однієї компанії або всередині групи споріднених компаній.

Міжнародні стандарти – найбільш повні системи, що містять, крім опису вимог до управління проєктами, навчання, тестування, аудит, консалтинг та інші елементи. Всеохопних міжнародних стандартів управління проєктами поки не існує, але найбільш відомі нормативні документи розглянемо детальніше [15].

Project Management Body of Knowledge (PMBOK) американського інституту управління проєктами (Project Management Institute – PMI). Цей стандарт оновлюється приблизно один раз в чотири роки. Одна з найбільш поширених редакцій датується 2000 р., а найактуальніша, четверта, версія стандарту – The Guide to the PMBOK, 7th Edition – вийшла в 2021 р. Стандарт був спочатку прийнятий Американським національним інститутом стандартів (ANSI) як національний стандарт в США, а в даний час знайшов світове визнання [1, 15].

В основі стандарту лежить процесний підхід до управління проєктами. Множина можливих процесів уявляється (подається) у вигляді тривимірного простору. По осях координат відкладаються ті вимірювання, які згадуються в рамкових стандартах. Можуть бути запропоновані й інші, наприклад, рівні управління, календарні періоди тощо. Кожна точка цього

простору є елементарним процесом управління. Наприклад, «планування ризиків на стадії впровадження системи».

Вибрані елементарні процеси утворюють процедури управління проєктами, які можуть бути побудовані за «осьовим» принципом.

Стандарт містить узагальнені принципи та підходи, які використовуються у сфері проєктного менеджменту, формалізовані і структуровані таким чином, щоб їх можна було використовувати в більшості проєктів у більшості випадків. Детально описуються дев'ять областей знань, пов'язаних з управлінням проєктами [1]:

- управління інтеграцією проєкту (Project Integration Management);
- управління змістом проєкту (Project Scope Management);
- управління термінами проєкту (Project Time Management);
- управління вартістю проєкту (Project Cost Management);
- управління якістю проєкту (Project Quality Management);
- управління людськими ресурсами проєкту (Project Human Resource Management);
- управління взаємодією в проєкті (Project Communications Management);
- управління ризиками проєкту (Project Risk Management);
- управління контрактами проєкту (Project Procurement Management).

Кожна галузь знання містить в собі окремі процеси, що виконуються менеджером при реалізації проєкту на тому чи іншому етапі. Процесно орієнтований підхід в управлінні проєктами, який використовується в стандарті, передбачає чіткий, формальний опис вхідних документів і даних, необхідних менеджеру для реалізації процесу, методів і засобів, які він може використовувати при його реалізації, і переліку вихідних документів процесу.

IPMA Individual Competence Baseline (IPMA ICB) є міжнародним нормативним документом, що визначає систему міжнародних вимог до компетентності менеджерів проєктів. Цей стандарт розроблений міжнародною асоціацією IPMA (International Project Managers Association). На його основі проводиться розробка національних систем вимог до компетентності фахівців в країнах, які є членами IPMA. Національні системи вимог мають відповідати IPMA ICB і офіційно затверджуватися (ратифікуватися) відповідними уповноваженими органами IPMA. Для 32 країн-членів IPMA він є основою для розробки національних збірок знань [1, 15].

ICB, на відміну від PMBOK, дотримується більш компетентного та дієвого підходу, зокрема визначає сфери кваліфікації та компетентності в управлінні проєктами, а також принципи оцінювання кандидата на отримання сертифіката. ICB містить 42 елементи (28 основних і 14 додаткових), що визначають вимоги до знань, майстерності та професійного досвіду в менеджменті проєктів.

ICB видано англійською, німецькою та французькою мовами. Основою для нього послужило кілька національних розробок: Body of Knowledge of APM (Великобританія); Beurteilungsstruktur, VZPM (Швейцарія); PM-Kanon, PM- ZERT / GPM (Німеччина); Criteres d'analyse, AFITEP (Франція).

Стандарт ISO 10006. Звернення до питань ефективності проєктного управління об'єктивно виявило гостру потребу в розробці системи управління якістю проєкту. При цьому особливе значення, поряд з вимогами до якості кінцевого продукту, стало надаватися якості процесів проєкту, відсутність належної уваги до яких призводила до не менш значущих негативних наслідків безпосередньо для створюваного продукту [1].

Стандарт ISO 10006 є основоположним документом із серії стандартів розглянутого профілю, підготовленим технічним комітетом ISO / TC 176 «Управління якістю і забезпечення якості» Всесвітньої федерації національних органів стандартизації (члени ISO).

Основний акцент зроблений на принципі ефективності проєктування оптимального процесу та на контролі цього процесу, а не на контролі кінцевого результату.

У цій серії стандартів процеси згруповані у дві категорії. До першої категорії віднесені процеси, пов’язані з забезпеченням продукту проєкту (проєктування, виробництво, перевірка). Опису останніх присвячений стандарт ISO 9004-1. Друга категорія охоплює безпосередньо процеси управління проєктом, а її представником є стандарт ISO 10006.

Даний стандарт охоплює десять груп процесів управління проєктом. Перша група являє процес розробки стратегії, який фокусує проєкт на задоволення потреб замовника і визначає напрямок ходу робіт. Друга група охоплює управління взаємозв’язками процесів. Решта вісім груп – це процеси, пов’язані з проєктним завданням, термінами, витратами, ресурсами, кадрами, інформаційними потоками, ризиком і матеріально-технічним постачанням (закупівлями).

Міжнародний стандарт ISO 10006 орієнтований на проєкти найширшого спектра – малі і великі, короткострокові і довгострокові, для різних навколишніх умов. Він не залежить від типу проєктованого продукту (це можуть бути технічні засоби, програмне забезпечення, напівфабрикати, послуги або їх поєднання). Це означає, що закладені в ньому рамкові вимоги вимагають подальшої адаптації даного керівництва до конкретних умов розробки та реалізації окремого проєкту [1].

Стандарт запозичує ключові означення з ISO 8402, зокрема такі терміни, як проєкт, продукт проєкту, план проєкту, учасник проєкту, процес, оцінювання ходу робіт. Для всіх процесів управління проєктом (планування, організація, моніторинг і контроль) застосовуються процеси і завдання менеджменту якості [15].

Стандарти зрілості управління проєктами теж набувають функції міжнародних. У 2004 р. був випущений стандарт оцінювання рівня зрілості організації з управління проєктами OPM3 (Organization Project Management Maturity Model), що містить методологію визначення стану управління проєктами в організації.

Термін «організаційна зрілість з управління проєктами» описує здатність організації відбирати проєкти та управляти ними таким чином, щоб це максимально ефективно підтримувало досягнення стратегічних цілей компанії.

OPM3 – це стандарт, який являє собою всебічний підхід, що допомагає організаціям оцінювати і розвивати свої можливості щодо ефективної реалізації проєктів. Він є свого роду ключем до організаційної зрілості управління проєктами і містить три взаємозалежних елементи [1]:

– елемент «знання» (knowledge) являє собою сотні кращих практик з управління проєктами, що характеризують ті чи інші рівні організаційної зрілості управління проєктами;

– елемент «оцінка» (assessment) є інструментом, що допомагає організаціям оцінити поточну зрілість управління проєктами та визначити галузі поліпшення;

– якщо організація приймає рішення розвивати практики управління проєктами і переходити на нові більш високі рівні зрілості, то в справу вступає елемент «покращення» (improvement), який допомагає компаніям побудувати схему розвитку управління проєктами таким чином, щоб забезпечити максимально ефективне досягнення своїх стратегічних цілей.

Основне призначення OPM3 – бути стандартом для корпоративного управління проєктами та організаційної зрілості з управління проєктами.

Основна відмінна риса OPM3 – це наявність унікальної бази даних, яка містить сотні кращих практик, опис тисяч ключових факторів успіху, результатів та іншої інформації, що характеризує розвиток зрілості управління проєктами в організації.

OPM3 спроєктований таким чином, щоб бути легким в розумінні і використанні, масштабованим, гнучким і налаштованим на споживача. Ґрунтуючись на базі OPM3 як стандарті управління проєктами, організація може успішно перейти до такого стану, коли проєкти будуть досягати поставлених цілей в рамках бюджету, термінів і, що більш важливо, переслідуючи корпоративні стратегічні цілі [1].

Специфіка IT-проєктів знаходить відображення також у специфічній методології управління проєктами:

- Capability Maturity Model Integration (CMMI);
- Microsoft Solution Framework;
- Rational Unified Process.

Контрольні питання

1) Назвіть найважливіші положення при управлінні проєктами.

2) Назвіть основні принципи нової концепції управління.

3) Які особливості при класифікації ІТ-проєктів?

4) Назвіть особливості життєвого циклу проєкту?

5) Які сучасні підходи до аналізу власної ідеї доцільно використовувати на етапі початку проєкту?

6) Які основні вимоги до учасників проєкту?

7) Які ключові стейкхолдери проєкту?

8) Як підібрати учасників в новий проєкт?

9) Які існують форми організаційної структури?

10) Назвіть успішні стандарти управління проєктами?

2 ПРАКТИЧНІ АСПЕКТИ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ

2.1 Процеси управління проєктами

Розбиваючи життєвий цикл проєкту на фази з проміжними результатами, ми, тим самим, робимо його більш керованим, знижуючи ступінь невизначеності від фази до фази. Більш того, будь-яка фаза може вилитися в окремий підпроєкт. Наприклад, на фазі визначення концепції проєкту можуть знадобитися глибокі маркетингові дослідження ринку, що можна виділити в окремий проєкт маркетингового аналізу зі своїми власними фазами [1].

Для того, щоб провести проєкт по фазах до результату, потрібно виконати деякі серії дій. Причому в кожному проєкті виконуються схожі процеси, які не залежать від предметної області. Такими загальними для всіх проєктів процесами зі схожим змістом є ініціація, планування, виконання, управління і завершення проєкту. Їх взаємозв'язок показаний на рисунку 2.1. Стрілками вказані напрямки потоків інформації.

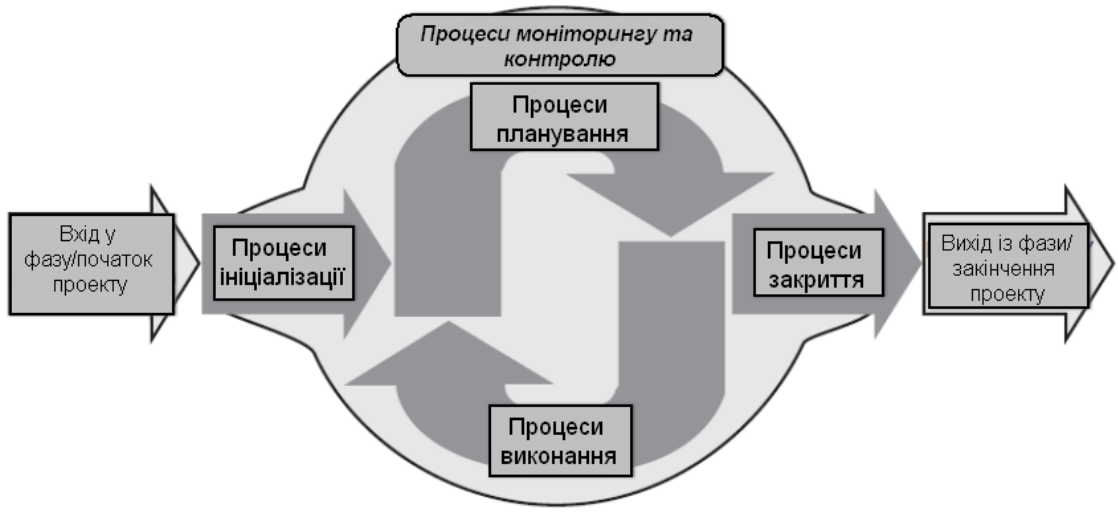


Рисунок 2.1 – Взаємозв'язок процесів управління

Як видно, за процесом ініціації проєкту слідують процеси планування і виконання проєкту. Процеси управління можуть повертатися до процесів планування, якщо не досягнуть кінцевого результату, що задовольняє цілі та мету проєкту. Процеси завершення закривають проєкт.

Характер управління проєктом вимагає, щоб група процесів моніторингу та контролю взаємодіяла з іншими групами процесів. Процеси моніторингу і контролю здійснюються в той же самий час, що і процеси, що входять в інші групи процесів. Таким чином, процес моніторингу та контролю зображений як «фонова» група процесів для інших чотирьох груп [1].

Ступінь взаємодії розглянутих груп процесів протягом життєвого циклу проєкту різна. У реальності вони накладаються один на одного, як показано на рисунку 2.2.

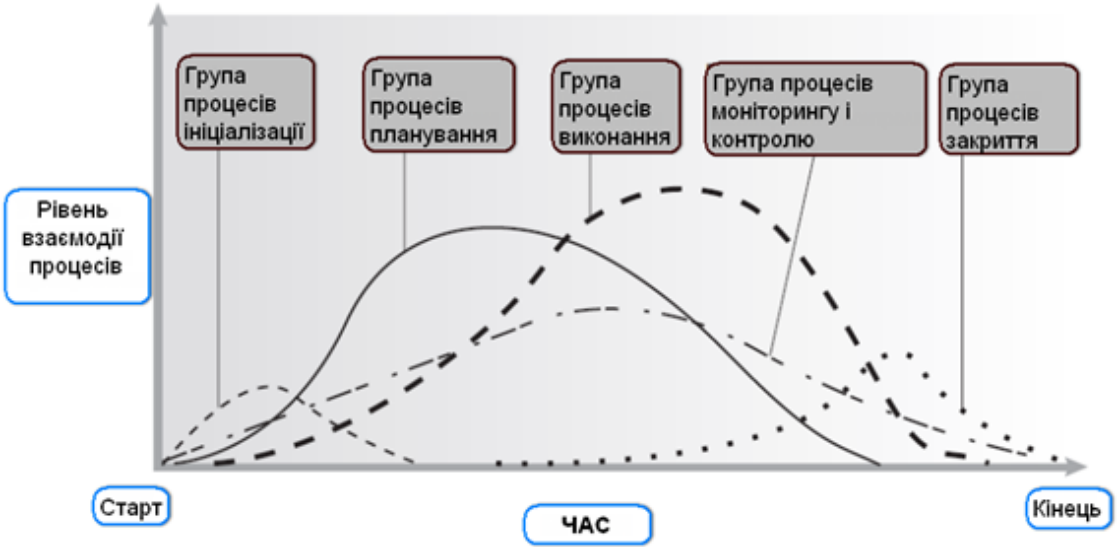


Рисунок 2.2 – Взаємодія груп процесів

Розкриємо зміст і взаємозв'язки кожної групи процесів. Групи процесів 2–5 покажемо схематично, як це прийнято в стандарті ANSI PMI PMBOK (рис. 2.3–2.6).

Охарактеризуємо зміст процесів управління проєктами.

Ініціація – визначення ділової потреби в проєкті і його авторизація, а саме [1, 6, 7]:

– вибір проєкту і визначення ділових потреб;
– збирання інформації;
– визначення цілей проєкту;
– визначення обмежень і припущень;
– розробка опису продукту;
– визначення обов'язків менеджера проєкту;
– визначення вимог до людських ресурсів (кадри, кваліфікація);
– оціночне визначення ресурсів;
– остаточне доопрацювання статуту проєкту і призначення менеджера проєкту.

Процеси планування спрямовані на розробку планів для складових проєкту (розклад, вартість, бюджет, якість, персонал, ризики, взаємодія, контракти тощо) і їх інтеграцію в цілісний, узгоджений документ – **План проєкту**. Як показано на рисунку 2.1, планування – це процес, що постійно повторюється протягом усього життєвого циклу проєкту [1].

У плануванні виділяють основні процеси, які присутні завжди і виконуються в чітко визначеному порядку, і допоміжні процеси, що залежать від характеру проєкту. На рисунку 2.3 показаний склад і зв'язки процесів планування.

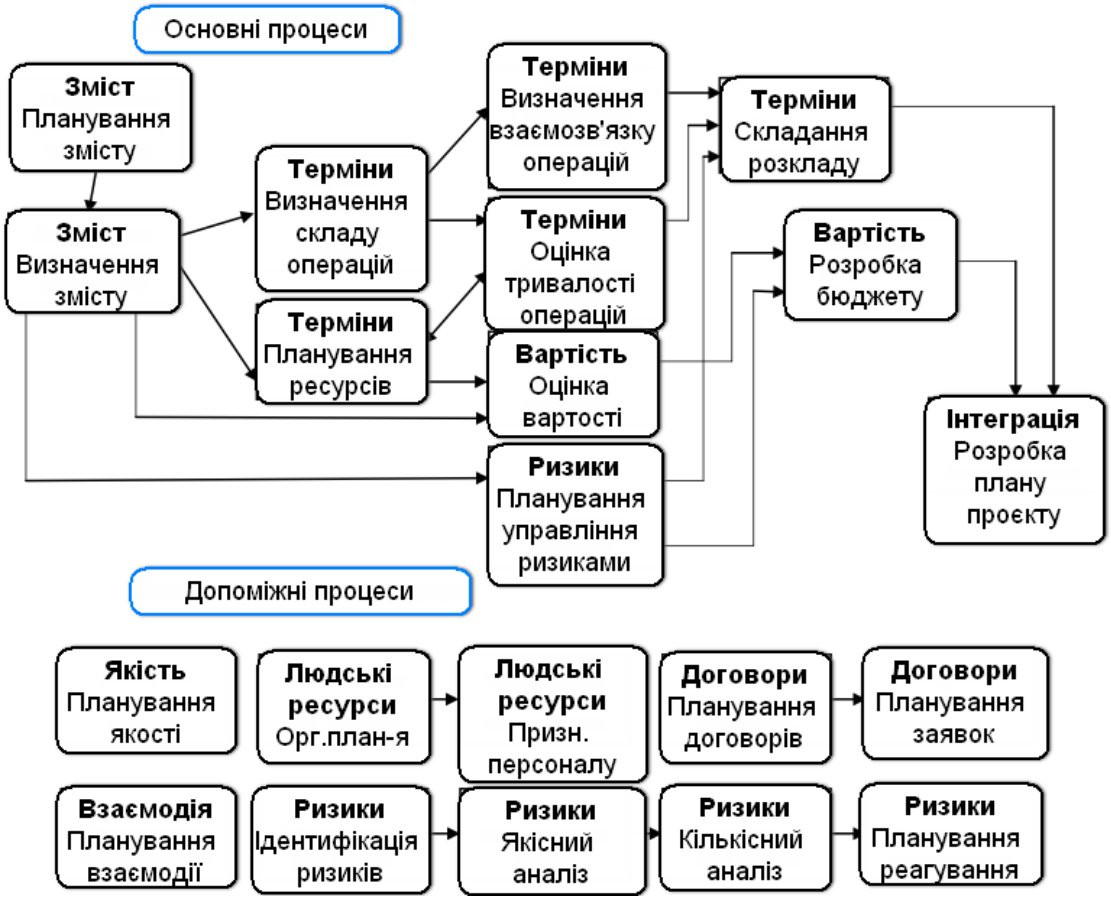


Рисунок 2.3 – Основні і допоміжні процеси планування

Планування змісту здійснюється на основі статуту проєкту і інших вхідних документів, таким чином складається документ, де міститься [1]:

а) уточнений опис продукту і результатів поставки;

б) класифікація можливих змін і спосіб їх виявлення;

в) порядок оцінювання та внесення змін в проєкт.

Визначення змісту – розбиття, декомпозиція цілей проєкту на менші і більш керовані частини (підцілі). Глибина декомпозиції має забезпечувати можливість призначення закінчених груп робіт і виконавців на ці частини. Результатом визначення змісту є ІСР (Ієрархічна Структура Робіт), англ. WBS (Work Breakdown Structure) [1].

Визначення складу операцій – підготовка переліку всіх операцій, що виконуються за проєктом, і уточнення ІСР. Операції, що не внесені до уточненої ІСР, вважаються внесеними в проєкт і не підлягають виконанню.

Планування ресурсів – визначення потреби (складу, кількості) в людських і матеріальних ресурсах, необхідних для виконання операцій проєкту.

Визначення взаємозв’язків операцій – виявлення взаємозв’язків і взаємозалежностей операцій, побудова мережних діаграм робіт проєкту. Частина операцій пов’язана між собою жорсткою логікою, інші операції можуть виконуватися в довільній послідовності, тобто пов’язані м’якою логікою [1].

Оцінювання тривалості операцій – встановлення кількості одиниць часу на операції проєкту, обчислення резервів часу і критичного шляху з мінімальною гнучкістю за часом.

Оцінювання вартості – кількісне оцінювання можливих витрат на ресурси, що залучаються, складання кошторису і плану управління вартістю.

Планування управління ризиками – встановлення підходу і заходів (коли, як і що робити) при загрозі або настанні небажаних і незапланованих подій і відхилень, з метою їх запобігання або ефективного реагування.

Складання розкладу – аналіз даних про послідовності і тривалості операцій, необхідні ресурси з метою створення розкладу виконання проєкту.

Розробка бюджету – визначення кошторисної вартості окремих пакетах робіт і проєкту в цілому.

Розробка плану проєкту – інтеграція даних попередніх процесів і складання узгодженого плану проєкту в одному або кількох документах.

Допоміжні процеси планування встановлюють стандарти якості, розподіл ролей і відповідальностей, інформаційні потреби учасників і способи взаємодії, виявляють ризики і наслідки їх впливу на цілі проєкту тощо.

Процеси виконання показані на рисунку 2.4. Виконання плану проєкту – це безпосереднє виконання його складових операцій. Допоміжні процеси забезпечують гарантії якості, комплектацію або розподіл робіт та інформації, проведення нарад про хід робіт і ідентифікацію змін, розвиток навичок і знань команди, збирання пропозицій постачальників, управління відносинами з постачальниками.

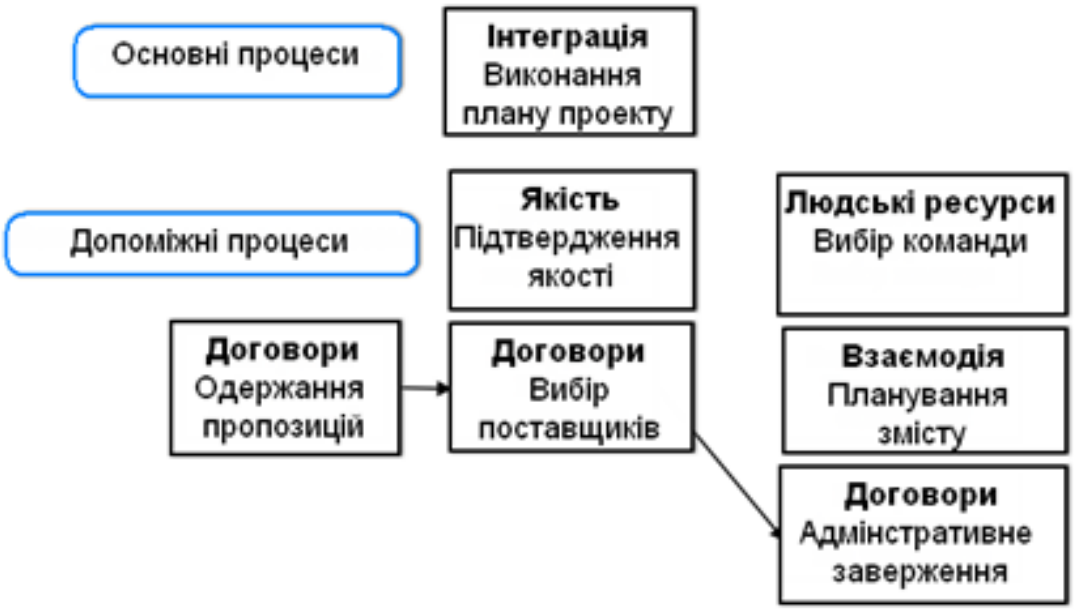


Рисунок 2.4 – Основні і допоміжні процеси виконання

Процеси моніторингу, управління, аналізу спрямовані на збирання і розподіл звітності стану проєкту, контроль відхилень, координацію змін

розкладу та бюджету. Склад і зв'язки процесів виконання показані на рисунку 2.5.

Рисунок 2.5 – Основні і допоміжні процеси моніторингу, управління, аналізу

Звітність з виконання виконання – це збирання та поширення інформації про хід проєкту і прогнози.

Загальне управління змінами – це координація змін у проєкті в цілому.

Допоміжні процеси забезпечують підтвердження правильності виконання робіт, фіксацію і прийняття змін, контроль і зміну розкладу та бюджету, відповідність стандартам якості та усунення причин її зниження, відстеження та виявлення ризиків, оцінювання заходів зниження ризиків [1, 13].

Процеси завершення впорядковують закриття проєкту і складаються з процесів закриття контрактів (договорів) і адміністративного завершення (рис. 2.6), а саме [1]:

– перевірка і тестування кінцевого продукту;
– остаточні розрахунки з усіма учасниками проєкту, фінансове закриття;
– остаточне оновлення документів проєкту;
– завершення звіту про виконання проєкту;
– збирання, інтеграція накопичених знань і формування архіву проєкту;
– офіційне приймання проєкту замовником, передача і запуск в експлуатацію;
– звільнення задіяних ресурсів.

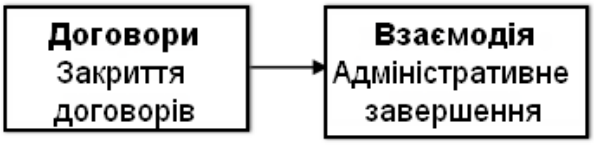


Рисунок 2.6 – Процеси завершення

Грамотне та ефективне виконання перерахованих процесів вимагає від менеджера проєкту знань в таких сферах [1]:

- Управління інтеграцією проєкту;
- Управління вартістю проєкту;
- Управління змістом проєкту;
- Управління термінами проєкту;
- Управління ризиками проєкту;
- Управління людськими ресурсами проєкту;
- Управління закупівлями проєкту;
- Управління комунікаціями проєкту;
- Управління якістю проєкту;
- Управління зацікавленими особами.

2.2 Моделі управління проєктами

Традиційна каскадна модель (англ. Waterfall model, іноді перекладають як модель «Водоспад») – модель процесу розробки програмного забезпечення, в якій процес розробки виглядає як потік, що послідовно проходить фази аналізу вимог, проєктування, реалізації, тестування, інтеграції та підтримки. Як джерело назви часто вказують статтю, опубліковану У. У. Ройсом (W. W. Royce) в 1970 році; при тому, що сам Ройс використовував ітеративну модель розробки [1, 16].

Те, що зараз називають «каскадна модель» ще у 1970 році Ройс описав в своїй статті у вигляді концепції. Там же він і обговорював недоліки цієї моделі і показав, як ця модель може бути доопрацьована до ітеративної моделі.

В оригінальній каскадній моделі Ройса, фази йшли в такому порядку [1]:

- визначення вимог;
- проєктування;
- конструювання (також «реалізація» або «кодування»);
- втілення;
- тестування та налагодження (також «верифікація»);
- інсталяція;
- підтримка.

Перехід від однієї фази до іншої можливий тільки після повного і успішного завершення попередньої

Розробник, дотримуючись каскадної моделі, переходить від однієї стадії до іншої тільки послідовно. Спочатку має завершитись етап «визначення вимог», в результаті якого отримується список вимог щодо програмного забезпечення. Після того як вимоги повністю визначені, відбувається перехід до процесу проєктування, в ході якого створюються документи, в яких міститься детальна інформація для програмістів, яка описує спосіб і план реалізації зазначених вимог. Після того як повністю виконане проєктування, програмістами реалізується отриманий проєкт. Наступна стадія процесу містить інтеграцію окремих компонентів, що розроблялись різними колективами програмістів. Після того як реалізація та інтеграція завершуються, відбувається тестування і налагодження програмного продукту. На цій стадії відбувається усунення всіх недоліків, які могли з'явитись на попередніх стадіях розробки. Далі програмний продукт впроваджується і забезпечується його постійна підтримка, тобто внесення, за необхідності, нової функціональності та усунення помилок [1, 16].

Тим самим, каскадна модель (рис. 2.7) говорить, що перехід від однієї фази розробки до іншої відбувається послідовно, тільки тоді, коли повністю успішно завершиться попередня фаза, а також, що переходів по фазах назад або вперед чи «перекриття» фаз не відбувається.

Потрібно відмітити, що також існують модифіковані каскадні моделі (зокрема модель самого Ройса), які мають невеликі або навіть значні варіації описаного процесу.

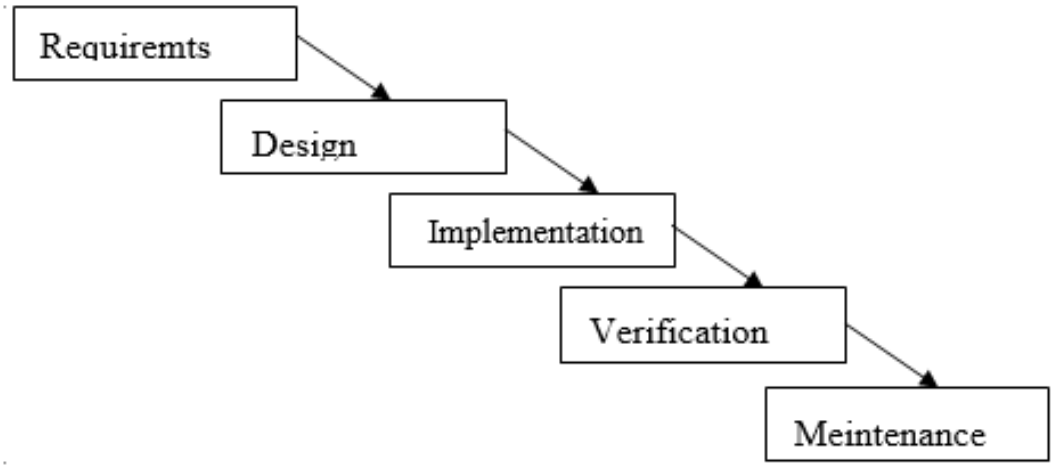


Рисунок 2.7 – Каскадна модель управління проєктами

Методику «Каскадна модель» досить часто критикують за недостатню гнучкість і формальне управління проєктом, що негативно відображається на термінах, вартості та якості. Однак при управлінні великими проєктами формалізація часто є дуже великою цінністю, тому що може кардинально знизити багато ризиків проєкту і зробити його більш прозорим.

Тому навіть в PMBOK 3-ої версії формально була закріплена лише методика «Каскадної моделі» і не були запропоновані альтернативні варіанти [1].

Починаючи з PMBOK 4-ої версії вдалося досягти компромісу між методологами, схильними до формального та поступального управління проєктом, і методологами, що роблять ставку на гнучкі ітеративні методи. Таким чином, починаючи з 2009 року, формально Інститутом управління проєктами пропонується як стандарт гібридний варіант методології управління проєктами, що поєднує в собі як плюси методики «Водоспаду», так і досягнення ітеративних методологів [16].

PRojects IN Controlled Environments 2 (PRINCE2) являє собою **структурований метод управління проєктами**, схвалений урядом Великобританії як стандарт управління проєктами в соціальній сфері. Методологія PRINCE2 містить в собі підходи до менеджменту, контролю та організації проєктів.

Спочатку метод був розроблений в 1989 році Central Computer and Telecommunications Agency (CCTA) у Великобританії як стандарт для керівництва проєктами в сфері інформаційних технологій. В даний час широко використовується і є «de facto» стандартом для керівництва проєктами у Великобританії.

Наведемо переваги даного підходу. PRINCE2 є структурованим підходом до управління проєктами, тобто це метод для управління проєктами в рамках чітко визначеної структури. PRINCE2 описує процедури для координації діяльності команди проєкту при розробці проєкту та контролю за ним. Також PRINCE2 описує процедури, які відбуваються при зміні проєкту, або коли початковий план зазнає істотних відхилень. У методі кожен процес визначається зі своїми основними входами і виходами, і з конкретними цілями і заходами, які будуть здійснюватися, в результаті чого ми отримуємо автоматичний контроль за будь-якими відхиленнями від плану. За рахунок поділу процесів на керовані етапи метод дає можливість ефективного управління ресурсами [1].

Як недолік можна зазначити відсутність будь-якого регламентування з боку методології підходів до управління контрактами поставок, учасниками проєкту, а також іншими процесами, які були винесені розробниками проєкту за рамки. Вважається, що кожен менеджер проєкту має обирати власні методи і підходи до цієї роботи.

Діаграма (рис. 2.8) показує процеси методу PRINCE2. Стрілки вказують напрямок інформаційних потоків.

Початок проєкту – як від початкової ідеї проєкту (відображеної в мандаті на проєкт) перейти до безпосередньої реалізації цих ідей. Створюється організація – мінімум призначаються керівник проєкту і Голова Комітету проєкту. Формується короткий опис проєкту (project brief) і підхід до його реалізації. Детально планується стадія запуску проєкту.

Рисунок 2.8 – Діаграма процесів методу PRINCE2

Ініціація проєкту – проводиться планування проєкту, зокрема і плану якості. Створюється економічне обґрунтування проєкту (Business Case) і відкривається журнал ризиків, проводиться оцінювання ризиків проєкту. Плануються етапи та точки контролю проєкту [1].

Управління проєктом – тут зосереджені шляхи прийняття рішень Комітетом проєкту (зокрема і дострокового завершення проєкту) та ситуаційне управління значними проблемами і відхиленнями.

Контроль стадій. Безпосередня робота керівника проєкту зі щоденного управління проєктом – видача і приймання завдань, фіксація складнощів і ризиків, прийняття рішення про ескалацію, звітність перед Комітетом.

Управління виробництвом продукту. Заходи, які виконавці та робочі групи мають вжити для визначення обсягів роботи, звіти про прогрес і передачу виконаної роботи [1].

Контроль меж (кордонів) стадій. Тут відбувається аналіз виконання плану стадії, проміжне планування наступної стадії, складання запасних планів, огляд ризиків та бізнес-плану. Служить для переходу між стадіями.

Завершення проєкту – як закрити проєкт, як управляти наступними діями, як розбирати огляди переваг проєкту.

Планування. Як планувати, незалежно від того, коли здійснюється планування.

Організаційна структура:

– Менеджер проєкту в традиційному розумінні;

– Комітет проєкту (project board), перед яким регулярно звітує менеджер. Складається з 3-х осіб – замовника, головного користувача і головного спеціаліста. Комітет проєкту відповідальний за прийняття стратегічних рішень. Менеджер проєкту зобов'язаний відстежувати можливі проблеми і пропонувати комітету альтернативні рішення. Комітет вирішує який шлях кращий [1];

– Служба проєктного офісу (project assurance), мета якої надавати незалежну думку про проєкт з точки зору тих же трьох груп людей – замовників, користувачів і фахівців (в предметній області). Служба готує три звіти [1]:

1) business report – звіт про фінансовий стан проєкту і вигідність проєкту в цілому;
2) user report – наскільки добре виконуються вимоги користувачів;
3) technical report – наскільки хороший проєкт в технологічному плані, чи туди він рухається;

– Служба адміністративної підтримки (адміністратори проєктів тощо) відповідальна за проведення зустрічей, доведення потрібної інформації до всіх її адресатів, збереження проєктної інформації тощо. У разі маленьких проєктів це робить менеджер проєкту.

При активному використанні моделі на практиці з'являються проблеми в переважній більшості IT-проєктів, за винятком окремих проєктів оновлення програмних систем для критично-важливих програмно-апаратних комплексів (авіація, медицина тощо).

У світі бізнес-систем каскадна модель не має застосовуватися, бо має бути висока динаміка коригування та уточнення, неможливо чітко визначити вимоги до початку робіт з реалізації (особливо для нових систем) через швидку мінливість вимог у процесі експлуатації.

В загальному випадку можна виділити такі *переваги* каскадної моделі:

– стабільність вимог протягом усього життєвого циклу розробки;
– можливість послідовного усунення нових проблем;
– визначеність і зрозумілість кроків моделі, простота застосування;
– спрощення планування, контролю та управління проєктом;
– доступність для розуміння замовниками;
– ефективність для проєктів з чіткими і зрозумілими, але важко реалізовними вимогами;
– ефективність для проєктів з надвимогами до якості при відсутності жорстких обмежень витрат і графіка робіт.

Серед *недоліків* цієї моделі можна виділити:

– складність чіткого формулювання вимог на початку життєвого циклу (ЖЦ) і неможливість їх динамічної зміни потім;

– послідовність лінійної структури процесу розробки, в результаті повернення до попередніх кроків для вирішення нових проблем дає збільшення витрат і порушення графіка робіт;

– проміжний продукт непридатний для використання;

– немає гнучкого моделювання систем, що не мають аналогів;

– пізнє виявлення проблем збирання через одночасну інтеграцію всіх результатів в кінці розробки;

– слабка участь користувача у створенні системи – тільки на початку (при розробці вимог) і в кінці (під час тестів і здачі);

– неможливість попереднього оцінювання якості системи користувачем;

– складність одноразового розподілу великих грошей.

Найефективніше використання цієї моделі може бути реалізоване при:

– розробці проєкту з чіткими, незмінними за ЖЦ вимогами, зрозумілою реалізацією і технічними методиками;

– розробці проєкту побудови системи або продукту, аналогічного до вже реалізованого розробниками раніше;

– розробці проєкту створення і випуску нової версії вже існуючого продукту або системи;

– розробці проєкту перенесення вже існуючого продукту на нову платформу;

– виконанні великих проєктів, в яких задіяно кілька великих команд розробників.

Гнучка методологія розробки (agile software development) – серія підходів до розробки програмного забезпечення, орієнтована на використання ітеративної розробки, динамічне формування вимог та забезпечення їх реалізації в результаті постійної взаємодії в рамках самоорганізованих робочих груп, що складаються з фахівців різних профілів. Існує кілька методик, які належать до класу гнучких методологій розробки, зокрема це й екстремальне програмування, DSDM, Scrum, FDD [1, 14].

Застосовується як ефективна практика організації праці невеликих груп (які роблять однорідну творчу роботу) в поєднанні з керуванням ними комбінованим (ліберальним і демократичним) методом.

Більшість гнучких методологій спрямована на мінімізацію ризиків, зводячи розробку до серії коротких циклів, які називаються ітераціями, що, зазвичай, тривають від двох до трьох тижнів. Кожна ітерація сама по собі виглядає як програмний проєкт у мініатюрі і містить в собі всі завдання, необхідні для мінізбільшення функціональності: планування, аналіз вимог, проєктування, програмування, тестування та документування. Хоча окремої ітерації, зазвичай, недостатньо для випуску нової версії продукту, мається на увазі, що проєкт гнучкого програмного забезпечення готовий до випуску в кінці кожної ітерації. Наприкінці кожної ітерації команда повторно оцінює пріоритети розвитку [1].

Agile-техніки наголошують на прямому спілкуванні віч-на-віч. Більшість agile-команд розташовані в одному офісі. Принаймні, в офісі є «замовники» (англ. product owner – замовник або його уповноважений представник, який визначає вимоги до товару; цю роль може виконувати менеджер проєкту, бізнес-аналітик або клієнт). У офіс також можуть входити тестувальники, дизайнери інтерфейсу, технічні редактори та менеджери.

Основною метрикою agile-методів є робочий продукт. Віддаючи перевагу прямій комунікації, agile-методи зменшують обсяг письмової документації порівняно з іншими методами. Це призвело до критики цих методів як недисциплінованих.

У лютому 2001 в штаті Юта США був випущений «Маніфест гнучкої методології розробки програмного забезпечення». Він був альтернативою таким «великоваговим» практикам розробки програмного забезпечення, як «метод водоспаду», який був золотим стандартом розробки в той час. Даний маніфест був схвалений і підписаний представниками методологій: екстремального програмування, Crystal Clear, DSDM, Feature driven development, Scrum, Adaptive software development, Pragmatic Programming. Гнучка методологія розробки використовувалася багатьма компаніями і до прийняття маніфесту, однак входження Agile-розробки в маси відбулося саме після цієї події [1].

Agile — це сімейство процесів розробки, а не єдиний підхід до розробки програмного забезпечення, який визначається Маніфестом Agile. Agile не містить в собі практику, але визначає цінності та принципи, якими керуються успішні команди.

Agile Manifesto розроблений і прийнятий 11–13 лютого 2001 року на лижному курорті The Lodge at Snowbird в горах Юти. Agile Manifesto містить 4 основні ідеї та 12 принципів. Потрібно відмітити, що Agile Manifesto не містить практичних порад.

Основні ідеї [1]:

– люди та взаємодія важливіші за процеси та інструменти;

– робочий продукт важливіший за вичерпну документацію;

– співпраця з замовником важливіша за узгодження умов договору;

– готовність до змін важливіша за дотримання початкового плану.

Принципи, які роз’яснює Agile Manifesto:

– задовольнити клієнта шляхом своєчасної та безперебійної доставки цінного програмного забезпечення;

– готовність змінювати вимоги навіть наприкінці розробки (це може підвищити конкурентоспроможність продукту);

– часте постачання працюючого програмного забезпечення (щомісяця, тижня або навіть частіше);

– тісне, щоденне спілкування замовника з розробниками протягом усього проєкту;

– до проєкту залучені мотивовані особистості, яким забезпечені необхідні умови праці, підтримка та довіра;

– рекомендований спосіб передачі інформації – особиста бесіда (віч-на-віч);

– працююче програмне забезпечення – найкращий показник прогресу;

– спонсори, розробники та користувачі мають мати можливість підтримувати постійний темп безперервно;

– постійна увага до вдосконалення технічних навичок та зручного дизайну;

– простота – мистецтво не виконувати зайву роботу;

– найкращі технічні вимоги, дизайн та архітектура – від самоорганізованої команди;

– постійне пристосування до мінливих обставин.

Одним із повторюваних моментів критики є те, що agile-підхід часто нехтує створенням плану («дорожньої карти») розробки продукту, а також управлінням вимогами, в процесі якого формується така «карта». Agile-підхід до управління вимогами не передбачає далекосяжних планів (насправді, управління вимогами просто не існує в цій методології), але передбачає здатність замовника раптово й несподівано в кінці кожної ітерації висувати нові вимоги, часто всупереч архітектурі доставленого товару. Це іноді призводить до катастрофічних «надзвичайних ситуацій» з масовим рефакторингом та змінами майже кожної ітерації [1].

Крім того, вважається, що робота в Agile мотивує розробників вирішувати всі завдання максимально простим і швидким способом, часто ігноруючи коректність коду з точки зору вимог платформи (підхід – «працює добре – і відмінно»). При цьому не враховується, що він може перестати працювати при найменших змінах або давати дефекти, що важко усунути після фактичного розгортання на клієнті. Це призводить до зниження якості продукту та накопичення дефектів. Одним із найпопулярніших гнучких підходів сьогодні є Scrum. Scrum – це авторська гнучка методологія розробки з нестандартним розподілом ролей у команді та унікальною організацією ітерацій. Scrum, як і інші agile-методики управління проєктами, відрізняється командним підходом, короткими ітераціями та безперервним удосконаленням. Ці принципи реалізуються через набір спеціальних ролей, правил, процесів та інструментів, які дозволяють командам виробляти продукт удвічі швидше.

Переваги Scrum полягають у нижчевикладеному.

1. *Прозорість*. Команда відкрита для обміну інформацією, знаннями, проблемами, кожен почувається залученим до досягнення спільної мети. Замовник завжди знає про робочий процес, вносить зміни, отримує достовірну інформацію про терміни здачі.

2. *Автономія команд*. Члени команди самі вирішують, як працювати над проєктом, свобода дій та відповідальність мотивують. Замовник передає вимоги команді безпосередньо, не використовуючи посередників.

3. *Мотивація за результатом*. Концепція Scrum дозволяє кожному члену групи щодня бачити свої та загальні досягнення. Замовник отримує збільшення функціональності з кожною ітерацією.

4. *Мінімізація ринкових ризиків*. Команда оперативно реагує на зміни вимог до проєкту та не робить зайвої роботи. Замовник отримує те, що йому потрібне і що затребуване на ринку.

5. *Мінімізація фінансових ризиків*. На виправлення помилок та додавання функціональності йде мало часу та коштів, все вкладається в бюджет.

Недоліки Scrum в IT-проєктах:

1. Не підходить для проєктів з туманними вимогами до кінцевого продукту, тому що замовник може нарощувати функціонал до нескінченності;

2. Складно навчитися правильно розставляти пріоритети та оцінювати задачі;

3. Успіх проєкту занадто залежить від Scrum-майстра;

4. Scrum складно використовувати у великих проєктах, необхідно масштабувати методологію та запровадити мітинги «scrum of scrums». У таких мітингах беруть участь представники кількох Scrum-команд, які працюють над суміжними продуктами.

На даний час є досить велика кількість програмних продуктів, що більшою чи меншою мірою підтримують підходи методології Scrum. Прикладом такого сервісу є Scrumdo (рис. 2.9) – «скрамовський» сервіс з плануванням ітерацій та картками завдань. Також даний сервіс містить стандартний пакет функцій зі звітами, діаграмами і гістограмами.

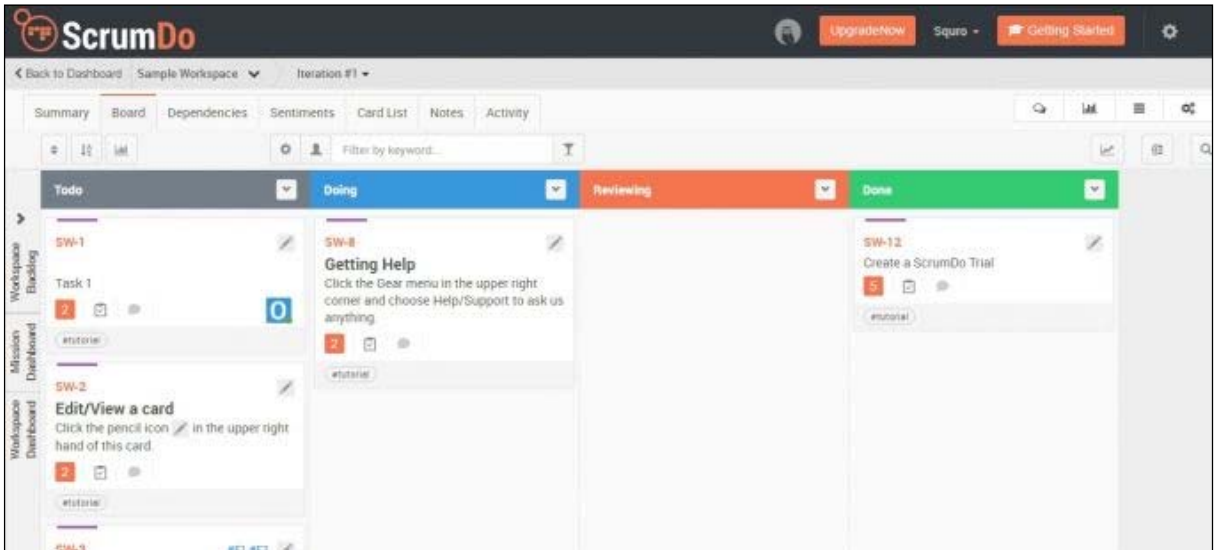


Рисунок 2.9 – Загальний вигляд інтерфейсу Scrumdo

RAD (Rapid application development – швидка розробка додатків) – концепція створення засобів розробки програмного забезпечення, в яких особлива увага приділяється швидкості та зручності програмування, ство-

рення технологічного процесу, що дозволяє програмістам швидко створювати комп'ютерні програми.

Концепція RAD стала відповіддю на такі недосконалі методи розробки програм у 1970-х та на початку 1980-х років, «водоспадна модель» (Waterfall model). Ці методи передбачали настільки повільний процес створення програми, що навіть вимоги до програми встигали змінитися до закінчення розробки. Засновником RAD вважається співробітник IBM Джеймс Мартін, який у 1980-х роках сформулював основні принципи RAD, ґрунтуючись на ідеях Баррі Бойма та Скотта Шульца. А в 1991 Мартін опублікував знамениту книгу, в якій докладно виклав концепцію RAD і можливості її застосування. В даний час RAD стає загальноприйнятою схемою створення інструментів розробки програмного забезпечення [1].

RAD передбачає, що розробка програмного забезпечення здійснюється невеликою командою розробників протягом приблизно 3-х–4-х місяців методом інкрементного прототипування з використанням засобів візуального моделювання та розробки. Технологія RAD передбачає активну участь замовника на ранніх етапах, а саме: обстеження організації, розробка системних вимог. Остання з цих властивостей передбачає повну відповідність вимогам замовника (як функціональним, так і не функціональним), з урахуванням їх можливих змін у процесі розробки системи, а також отримання якісної документації, що забезпечує простоту експлуатації та супроводу системи. Це означає, що додаткові витрати на підтримку відразу після поставки будуть значно меншими. Таким чином, загальний час від початку розробки до отримання прийнятного продукту під час використання даного методу значно скорочується [17].

Технологію RAD потрібно використовувати, коли чітко визначено деякі пріоритетні параметри розробки проєкту [1]:

1. Необхідно реалізувати проєкт у стислі терміни. Швидка реалізація проєкту дозволяє створити систему, що відповідає вимогам сьогодення. Якщо система розрахована на тривалий період, то, ймовірно, за цей час істотно зміняться фундаментальні положення, що регулюють діяльність організації, тобто система застаріє ще до закінчення періоду її проєктування;

2. Нечіткі вимоги до програмного забезпечення. У більшості випадків замовник має приблизне уявлення про майбутній програмний продукт і не може чітко сформулювати всі вимоги до програмного забезпечення. Вимоги можуть взагалі не бути визначені до початку проєкту або можуть змінюватися під час його реалізації;

3. Проєкт реалізується з обмеженим бюджетом. Розробка здійснюється невеликими RAD-групами в стислі терміни, що забезпечує мінімальні трудовитрати і дозволяє вписатися в бюджетні обмеження;

4. Інтерфейс користувача (GUI) є основним фактором. Немає сенсу змушувати користувача «малювати» картинки. Технологія RAD дозволяє продемонструвати інтерфейс в прототипі, і незабаром після запуску проєкту;

5. Можна розділити проєкт на функціональні компоненти. Якщо передбачається велика система, її потрібно розбити на менші частини, кожна з яких має різні функції. Вони можуть виконуватися послідовно або паралельно (в останньому випадку задіяно кілька груп RAD);

6. Низька обчислювальна складність програмного забезпечення.

Технологія RAD не є універсальною, тобто її використання не завжди доцільно. Наприклад, у проєктах, де вимоги до програмного продукту чітко визначені і не мають змінюватися, залучення замовника до процесу розробки не потрібне, а ієрархічна розробка (каскадний метод) може бути більш ефективною. Те ж саме стосується проєктів, складність яких визначається необхідністю реалізації складних алгоритмів, а роль і обсяг інтерфейсу користувача невеликі [1].

Принципи технології RAD спрямовані на забезпечення трьох її основних переваг – високу швидкість розробки, низьку вартість і високу якість. Досягти високої якості програмного забезпечення непросто, і одна з основних причин труднощів полягає в тому, що розробник і замовник порізному бачать предмет розробки (ПЗ) [17]:

– Набір інструментів має бути спрямований на мінімізацію часу розробки;

– Створення прототипу для уточнення вимог замовника;

– Циклічна розробка: кожна нова версія продукту базується на оцінці замовником результату попередньої версії;

– Мінімізація часу розробки версії за рахунок перенесення готових модулів і додавання функціональності в нову версію;

– Команда розробників має тісно співпрацювати, кожен член має бути готовим взяти на себе численні обов’язки;

– Управління проєктом має мінімізувати тривалість циклу розробки.

Принципи RAD (рис. 2.10) застосовуються за межами реалізації на всіх етапах життєвого циклу, охоплюючи опитування організації, формування вимог, аналіз та проєктування.

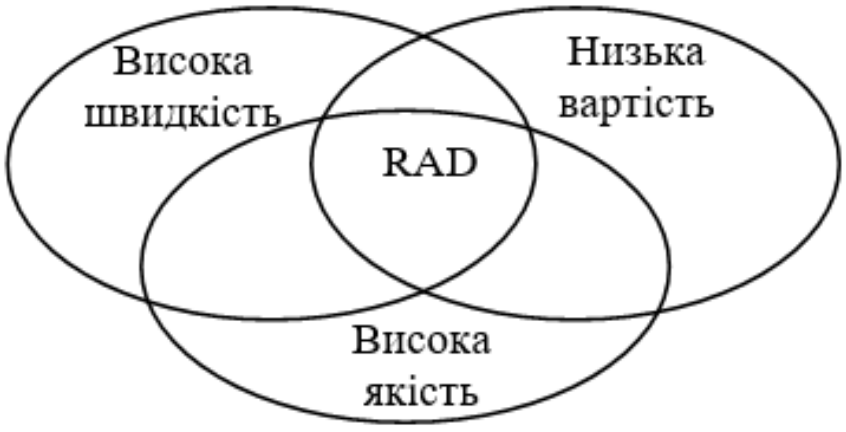


Рисунок 2.10 – Принципи RAD

Модель швидкої розробки RAD (рис. 2.11) складається з 4-х етапів:

– планування;
– проєктування;
– конструювання;
– перемикання.

Рисунок 2.11 – Модель швидкої розробки RAD

Планування – це набір вимог, отриманих від систематичного планування та аналізу процедури розробки життєвого циклу (SDLC). На цьому етапі користувачі, менеджери та IT-фахівці обговорюють цілі проєкту, його обсяг, системні вимоги та труднощі, які можуть виникнути під час розробки. Етап завершується узгодженням ключових моментів з групою RAD та отриманням дозволу від керівників проєкту на продовження роботи [1, 17].

Проєктування користувачем – на цьому етапі користувачі, взаємодіючи з системними аналітиками, розробляють моделі та прототипи, які містять всі необхідні функції системи. Команда RAD зазвичай використовує інструменти Unified Application Development (JAD) і CASE для перетворення прототипів користувачів у робочі моделі. Проєктування користувачем – це тривалий інтерактивний процес, який дозволяє користувачам зрозуміти, змінити та в кінцевому підсумку вибрати робочу модель, яка відповідає їхнім вимогам.

Конструювання – етап, на якому основним завданням є розробка програм і додатків. Подібна фаза «впровадження» є в SDLC. Однак користувачі продовжують брати участь у RAD і можуть запропонувати зміни або покращення у формі звітів, які вони розробили. У їхні завдання входять: програмування та розробка додатків, написання коду, інтеграція модулів та тестування системи.

Перемикання – охоплює операції перетворення даних, тестування, перехід на нову систему та навчання користувачів. Його завдання нагадує завершальний етап SDLC. Порівняно з традиційними методами розробки програмного забезпечення, весь процес займає багато часу. В результаті нова система з більшою ймовірністю буде побудована, доставлена замовнику та встановлена на робочому місці [1].

Технологія швидкої розробки додатків (RAD) забезпечує:

– швидкість просування програмного продукту на ринок;
– зручний інтерфейс;
– легка адаптація проєкту до мінливих вимог;
– простота розробки функціональності системи.

Спіральна (ризикова) модель (рис. 2.12) пропонує набір можливостей адаптації вдалих аспектів існуючих моделей процесів ЖЦ. Дана модель характерна при розробці новаторського (нетипового) ПЗ. На початку роботи у замовника і розробника немає чіткого бачення підсумкового ПЗ (вимоги не можуть бути чітко визначені) або стовідсоткової впевненості в успішній реалізації (ризики дуже великі).

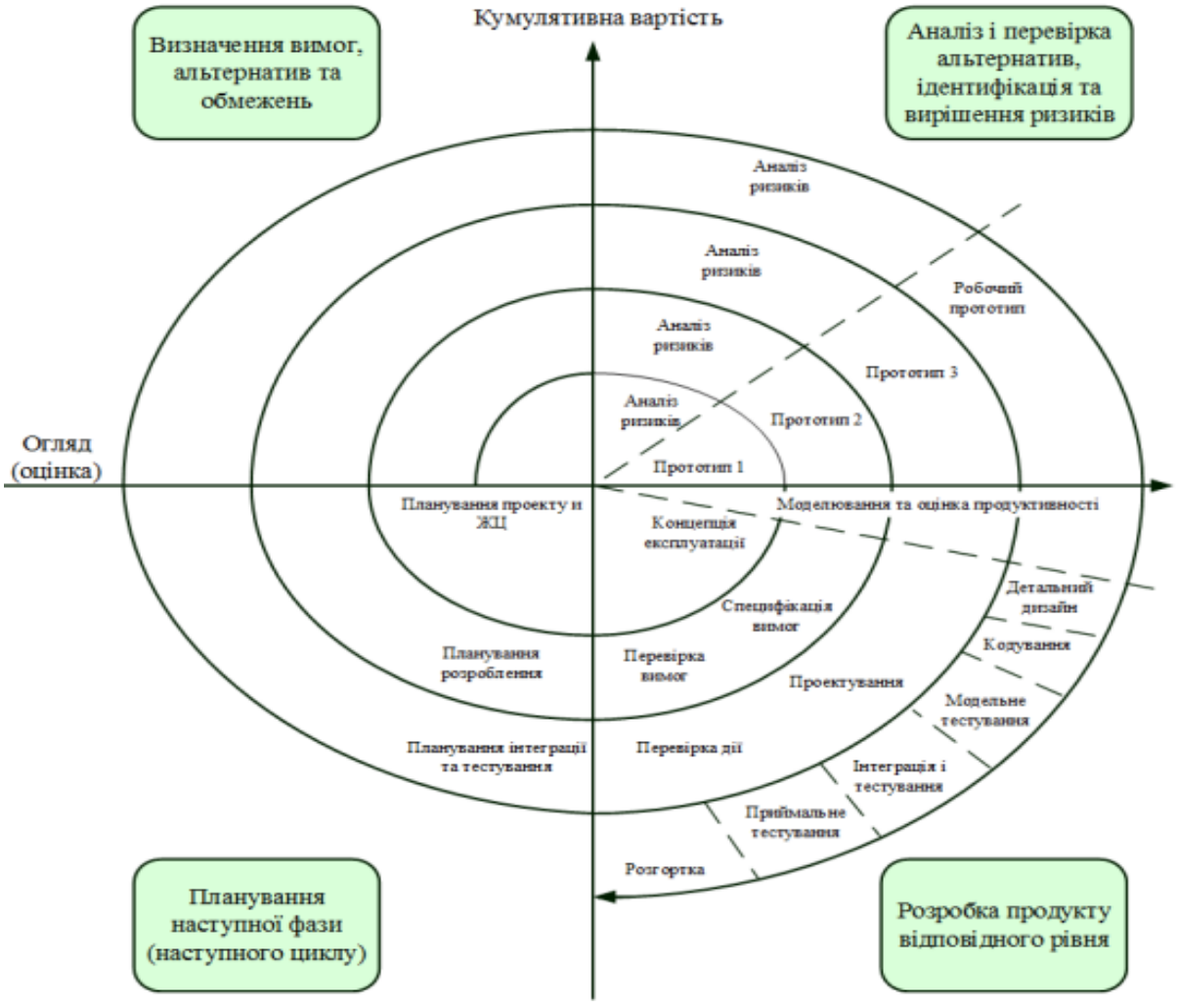


Рисунок 2.12 – Спіральна (ризикова) модель

У зв’язку з цим приймається рішення про розробку ПЗ по частинах з можливістю зміни вимог або відмови від подальшої розробки. Як видно з рисунка, розробка проєкту може бути завершена не тільки після фази впровадження, але й після фази аналізу ризиків.

Перевагами спіральної (ризикової) моделі є:

– користувач досить швидко бачить працююче програмне забезпечення, активізує процес уточнення та доповнення вимог;

– дозволяє змінювати вимоги до розробки інформаційних систем, що характерно для більшості розробок, зокрема стандартних;

– забезпечує більшу гнучкість в управлінні проєктами;

– дозволяє отримати більш надійне і стабільне програмне забезпечення. При розробці програмного забезпечення на кожній ітерації визначаються та виправляються помилки та недоліки;

– дозволяє покращити процес розробки – аналіз, що проводиться на кожній ітерації, дозволяє оцінити, що необхідно змінити в організації розробки і покращити це на наступній ітерації;

– ризики замовника знижуються – він може завершити розробку неперспективного проєкту з мінімальними втратами.

Недоліки спіральної (ризикової) моделі:

– росте невизначеність у розробника в перспективах розвитку проєкту;

– ускладнені операції тимчасового і ресурсного планування всього проєкту в цілому. Для вирішення цієї проблеми потрібно ввести тимчасові обмеження на кожну із стадій ЖЦ;

– перехід здійснюється за планом, навіть якщо не вся запланована робота виконана. План складається на основі статистичних даних з попередніх проєктів та особистого досвіду розробників.

2.3 Документація проєкту

Якщо Ви поговорите з досвідченим керівником проєкту, то він, напевно, дасть Вам чимало порад про те, що Вам варто, а чого не варто робити для успішного управління будь-яким проєктом. В основному всі вони будуть про те, як управляти людьми та їх роботою (а також і Вашою роботою), як надавати результати вчасно і згідно з бюджетом, при цьому тримати рівень ризику на мінімумі [1, 18].

На початку Вашого проєкту Вам знадобиться створити лише односторінковий документ – *статут проєкту*. Даний документ має забезпечити те, що Ви і Ваш клієнт розуміють основні цілі проєкту. Якщо Ви не знаєте, куди йдете, як же Ви зрозумієте, що Ви біля цілі?

Для отримання даної інформації Вам варто влаштувати зустріч з Вашим клієнтом і задати такі 3 важливих питання:

– Які цілі Вашого проєкту?

– Що Ви хочете виробляти або надавати?

– Яке бізнес-обґрунтування для виконання даного проєкту?

Після зустрічі запишіть всі відповіді на дані питання у Вашому статуті проєкту і перешліть його клієнту для отримання підтвердження.

Тепер Ви виконали найбільш важливу частину вашого проєкту, тобто усвідомили і погодили з Вашим клієнтом призначення Вашого проєкту.

Якщо Ви проаналізуєте час, витрачений на виконання даного кроку у Вашому проєкті, то у Вас вийде приблизно дві зустрічі і 30 хвилин для запису всієї інформації, тобто приблизно дві години на маленький проєкт. Не так вже й багато [18].

Тепер Ви можете створити другий документ – *план проєкту*. Він має містити в собі список всієї роботи, яка має бути виконана (іншими словами, масштаб проєкту), хто її виконуватиме, витрати і час на виконання даної роботи і, нарешті, простий огляд того, що може піти не так, або ж дослідження ризиків.

Зазвичай раз в тиждень або на місяць Вам потрібно надавати *звіт про прогрес* Вашим клієнтам. Вони хочуть знати, що було виконано, як і скільки часу і коштів було витрачено. Вони також хочуть знати чи потрібна Вам їх допомога у вирішенні проблем. У Ваші обов'язки входить збирання цієї інформації, щоб Ви могли створити даний документ.

Існує чотири основних типи документації на ПЗ [1]:

– архітектурна/проєктна – огляд програмного забезпечення, охоплюючи опис робочого середовища та принципи, які потрібно використовувати при створенні програмного забезпечення;

– технічна – документація на код, API, алгоритми, інтерфейси;

– для користувача – інструкції для кінцевих користувачів, адміністраторів системи та іншого персоналу;

– маркетингова.

Проєктна документація зазвичай описує продукт у загальних рисах. Не описуючи, як щось буде використано, відповідає на питання «Чому це так». Наприклад, у проєктному документі програміст може описати обґрунтування того, чому структури даних організовані так, як вони є. Він описує причини, чому клас, розроблений певним чином, висвітлює закономірності, а в деяких випадках навіть дає ідеї щодо того, як зробити покращення в майбутньому. Ніщо з цього не внесено в технічну або користувацьку документацію, але це дійсно важливо для проєкту [1].

При створенні програми одного коду зазвичай недостатньо. Має бути текст, що описує різні аспекти того, що робить код. Така **технічна документація** часто вноситься безпосередньо у вихідний код або постачається разом з ним.

Така документація є високотехнічною й, в основному, використовується для визначення та опису API, структур даних і алгоритмів.

Для створення технічної документації часто використовуються такі автоматизовані інструменти, як Doxygen, javadoc, NDoc та інші. Вони беруть інформацію зі спеціально створених коментарів у вихідному коді та

створюють довідкові інструкції в будь-якому форматі, наприклад, текстовому чи HTML.

Використання генераторів документації та коментарів до документації є корисним інструментом для багатьох програмістів з різних причин. Зокрема, при такому підході документація є частиною вихідного коду, і ті самі інструменти можна використовувати для компіляції програми та збору документації для неї одночасно. Це також полегшує оновлення документації [1].

На відміну від технічної документації, яка зосереджується на коді та як він працює, **документація користувача** лише описує, як користуватися програмою.

Якщо продукт являє собою програмну бібліотеку, то документація користувача та документація коду стають дуже близькими, майже еквівалентними поняттями. Але загалом це не так.

Зазвичай, призначена для користувача документація являє собою інструкцію користувача, яка описує кожну функцію програми, а також кроки, які потрібно виконати для використання цієї функції. Хороша для користувача документація йде ще далі й містить інструкції, що робити, якщо виникнуть проблеми. Дуже важливо, щоб документація не вводила в оману та була актуальною. Інструкція має мати чітку структуру, також важливі логічність і простота [18].

Існує три підходи до організації документації для користувача. Довідник з основних функцій (tutorial) – найбільш корисний для нових користувачів, він послідовно проводить по низці кроків, призначених для виконання певних типових задач. Тематичний підхід, при якому кожна глава інструкції присвячена якійсь окремій темі, більше підходить для користувачів, які вдосконалюються. В останньому, третьому, підході команди або завдання організовані у вигляді алфавітного довідника – часто це добре сприймається досвідченими користувачами, які добре знають, що вони шукають. Скарги користувачів зазвичай стосуються того, що документація охоплює тільки один з цих підходів, і тому добре підходить лише для одного класу користувачів [1].

У багатьох випадках розробники програмного продукту обмежують набір документації для користувача лише вбудованою системою допомоги (online help), що містить довідкову інформацію про команди або пункти меню. Робота з навчання нових користувачів і підтримки вже існуючих перекладається на приватні видавництва, які часто надають значну допомогу розробникам.

Маркетингова документація – для багатьох додатків, щоб зацікавити людей, привернувши їх увагу до продукту, потрібно розмістити рекламні матеріали поруч з ними. Ця форма документації спрямована на [1]:

– стимулювання інтересу до продукту з боку потенційних користувачів;

– інформування їх про те, що робить продукт, щоб їхні очікування збігалися з тим, що вони отримають;

– пояснення позиції продукту порівняно з рішеннями-конкурентами.

Одна з хороших маркетингових практик – надання слогана, тобто простої фрази, що легко запам'ятовується та ілюструє те, що необхідно донести до користувача, а також характеризує відчуття, яке створює продукт.

Часто трапляється, що слоган продукту та інші маркетингові матеріали дають більш чітке уявлення про можливості та способи використання програми, ніж будь-що інше.

2.4 Управління ризиками проєкту

Ризик – це така невизначена подія або умова, настання якої негативно або позитивно позначається на меті проєкту, як зміст, розклад, вартість і якість. Причому вплив ризику на проєкт може бути як негативним, так і позитивним. На практиці ризики розглядаються як загроза проєктам.

Ризик може бути викликаний однією або декількома причинами і в разі виникнення може вплинути на один або кілька аспектів. Причиною може бути існуюча або потенційна вимога, припущення, обмеження або умова, яка створює ймовірність негативних або позитивних наслідків. Наприклад, причиною ризику може бути необхідність отримання дозвільної документації в галузі охорони навколишнього середовища або недостатня кількість персоналу, залученого для розробки проєкту. Ризиком в першому випадку буде затримка з видачою дозволу органом контролю, а в другому, в разі сприятливої можливості, додатковий персонал, який може бути залучений до розробки проєкту, для призначення на проєкт. Виникнення будь-якого з цих точно не відомих заздалегідь подій може вплинути на проєкт, його зміст, вартість, розклад, якість або виконання. Умови ризику також можуть містити аспекти організації або середовища проєкту, які сприяють підвищенню ризику (наприклад, незрілі практики управління проєктами, відсутність загальних систем управління, одночасне виконання кількох проєктів або залежність від зовнішніх учасників проєкту, які неможливо безпосередньо контролювати [1].

Причини ризиків проєкту знаходяться у невизначеності, яка присутня в усіх проєктах. Відомі ризики – це ті ризики, які були ідентифіковані і проаналізовані, що дозволяє планувати реагування на них. Для тих відомих ризиків, якими неможливо управляти проактивно, потрібно виділити резерв на можливі втрати. Невідомими ризиками неможливо управляти проактивно, і, отже, для них можна виділити управлінський резерв. Існуючий негативний ризик проєкту розглядається як проблема. Можливість управління ризиками залежить від рівня визначеності (рис. 2.13) [1].

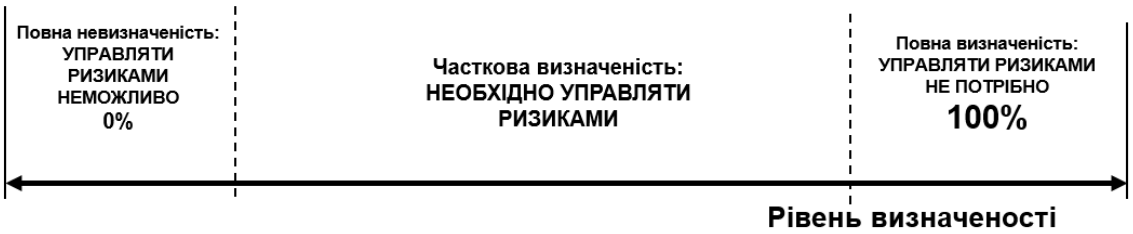


Рисунок 2.13 – Спіральна (ризикова) модель

Окремі ризики проєкту відрізняються від загального ризику проєкту. Загальний ризик проєкту відображає ефект невизначеності стосовно всього проєкту. Це більше ніж сума окремих ризиків в проєкті, оскільки сюди входять всі джерела невизначеності проєкту. Він відображає схильність зацікавлених сторін впливу (як позитивного, так і негативного) від варіацій в кінцевому результаті проєкту [1, 10].

Базові плани з утримання, щодо термінів і вартості створюються з урахуванням нейтралізації частини ризиків, передбачуваних з імовірністю 100%. Тому базові плани містять роботи і витрати, які належить виконати з імовірністю 100% [12].

Ризики, що не очікувані з імовірністю 100% і не припускають нейтралізації, не відображаються в базових планах. Настання таких ризиків призводить до наслідків, які, як правило, виливаються в додаткові роботи і витрати (грошей і часу), ймовірність яких до цього вважалася меншою 100%. Такі ризики покриваються за рахунок резервів [1].

Важливо враховувати динаміку зміни ризиків в ході реалізації проєкту (рис. 2.14).

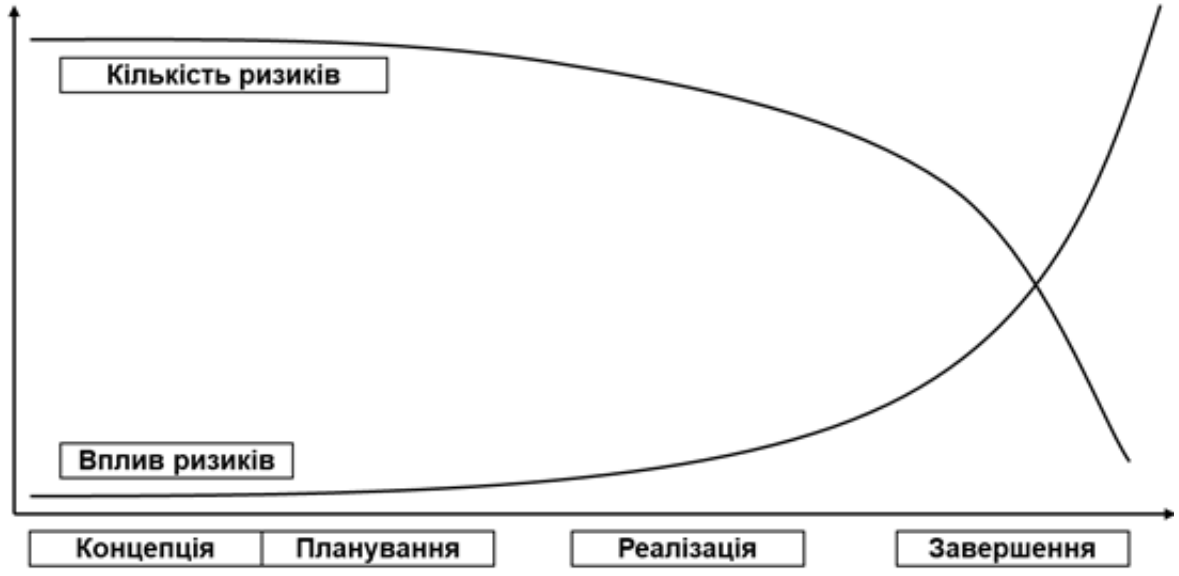


Рисунок 2.14 – Динаміка зміни ризиків в ході реалізації проєкту

Класифікації ризиків. Ризики поділяють на відомі і невідомі. Відомі ризики – це ризики, які виявлені, ідентифіковані. Їх можна аналізувати і планувати. Невідомі – неідентифіковані ризики, які не піддаються управлінню, але можуть бути враховані при формуванні резервів.

З точки зору керованості, ризики поділяють на внутрішні і зовнішні. Внутрішні ризики – це події, умови та процеси, які команда проєкту може контролювати. Зовнішні ризики – це події, умови та процеси, які виходять за межі впливу команди проєкту. Наприклад, зміни законодавства країни, зміни вимог і пріоритетів спонсорів, зміни в організації-виконавці або у замовника, ринкові зміни, цивільні і природні катаклізми та інші форс-мажорні обставини [10].

Залежно від джерел виникнення ризики можна розділити на такі категорії [1]:

– технічні ризики – пов'язані з помилками проєктування, використанням неперевірених технологій, порушенням галузевих стандартів тощо;

– управлінські ризики – пов'язані з недоліками в плануванні та управлінні проєктом на рівні керівника проєкту. Наприклад, невдалі графіки, погано описані ролі та обов'язки, підбір недостатньо кваліфікованого персоналу, часті зміни в команді, помилкові оцінки та розрахунки реалізації проєкту тощо;

– організаційні ризики – виникають через недоліки на рівні вищого керівника проєкту та пов'язані з неузгодженістю між проєктами, низькою проєктною дисципліною та конфліктами щодо ресурсів, несумісністю цілей проєкту, сильним впливом зовнішніх факторів, недостатнім або нестабільним фінансуванням тощо;

– бізнес-ризики – пов'язані зі змінами бізнес-середовища та умов ведення бізнесу, в яких було розпочато проєкт. Наприклад, помилки в прогнозах ринку, низька відповідальність провідних стейкхолдерів, зміна пріоритетів і вимог спонсора або замовника, інші ризики замовника тощо;

– ризики навколишнього середовища – природно-кліматичні, екологічні та інші ризики;

– соціальні і політичні – страйки, державні перевороти тощо;

– ризики зловмисних дій.

Процес виявлення та ідентифікації ризиків має бути безперервним і постійним протягом всіх фаз життєвого циклу проєкту. Адже ризики можуть змінюватися, зникати, можуть бути виявлені нові, раніше невідомі ризики. З просуванням проєкту до завершення і зниженням загальної невизначеності велика частина ризиків має бути ідентифікована, оцінена та усунена.

Вирішуючи завдання з управління ризиками, менеджер проєкту і команда управління ризиками проєкту періодично проходять і повертаються до таких дій [1]:

– ідентифікація та оцінювання ризиків (2 основних параметри оцінювання ризику – ймовірність виникнення і наслідки);

– мінімізація ймовірності настання ризиків, запобігання або підготовка до настання ризиків;

– реагування на ризики, мінімізація негативних наслідків ризиків, а якщо ризик все ж настав, то знаходження способів подолання та забезпечення виконання проєкту за планом;

– фіксація відхилень від базових планів щодо термінів, за вартістю і за змістом; якщо наслідки серйозні і відхилення від планів неминучі, зміна і узгодження планів з урахуванням наслідків, виконання коригувальних дій.

Таким чином, управління ризиками можна визначити як систематичний процес ідентифікації, аналізу і реагування на проєктні ризики.

Управління ризиками вимагає додаткових витрат. Ці витрати мають порівнюватися з бюджетом всього проєкту, бути необхідними і достатніми, щоб забезпечити досягнення цілей і результатів проєкту. Суми витрат на управління ризиками в проєктах можуть коливатися в діапазоні 1–15% всього бюджету проєкту [1,10].

Дії з управління ризиками проводяться на етапах планування та управління. Ці дії залежать від характеру проєкту і є допоміжними процесами. У стандарті ANSI PMBOK виділяють 6 складових процесів управління ризиками, причому перші п’ять з них спрямовані на попередню роботу над ризиками, на підготовку до виникнення ризиків [1]:

– Планування управління ризиками;
– Ідентифікація ризиків;
– Якісний аналіз ризиків;
– Кількісний аналіз ризиків;
– Планування реагування на ризики;
– Контроль ризиків.

Планування управління ризиками – процес, який визначає, яким чином здійснювати управління ризиками проєкту (рис. 2.15). Ключова вигода даного процесу полягає в забезпеченні того, щоб ступінь, тип і наочність управління ризиками були відповідні ризикам і важливості проєкту для організації [12].

Рисунок 2.15 – Динаміка зміни ризиків в ході реалізації проєкту

Процес планування управління ризиками має визначити правила і підходи до управління ризиками проєкту, а саме:

– способи виявлення та джерела ризиків;
– ролі і відповідальності як на рівні керівництва, так і на рівні членів команди – може бути створена спеціальна команда управління ризиками;
– бюджет управління ризиками;
– періодичність ідентифікації, аналізу та оцінювання ризиків;
– критерії порогових величин, після яких потрібне втручання;
– форми звітності і документування.

Розкриття перерахованих пунктів приводить до створення на виході процесу плану управління ризиками. На вході процесу потрібно мати такі джерела, як [1]:

– план управління проєктом;
– статут проєкту;
– реєстр зацікавлених осіб;
– фактори середовища підприємства;
– активи процесів організації;
– інша необхідна та доступна інформація з плану проєкту.

Корисним додатком до плану управління ризиками може бути побудова ієрархічної структури ризиків (ІС Ризиків). Джерела (причини) ризиків деталізуються до таких ризиків, за які можна призначити одну відповідальну особу (рис. 2.16).

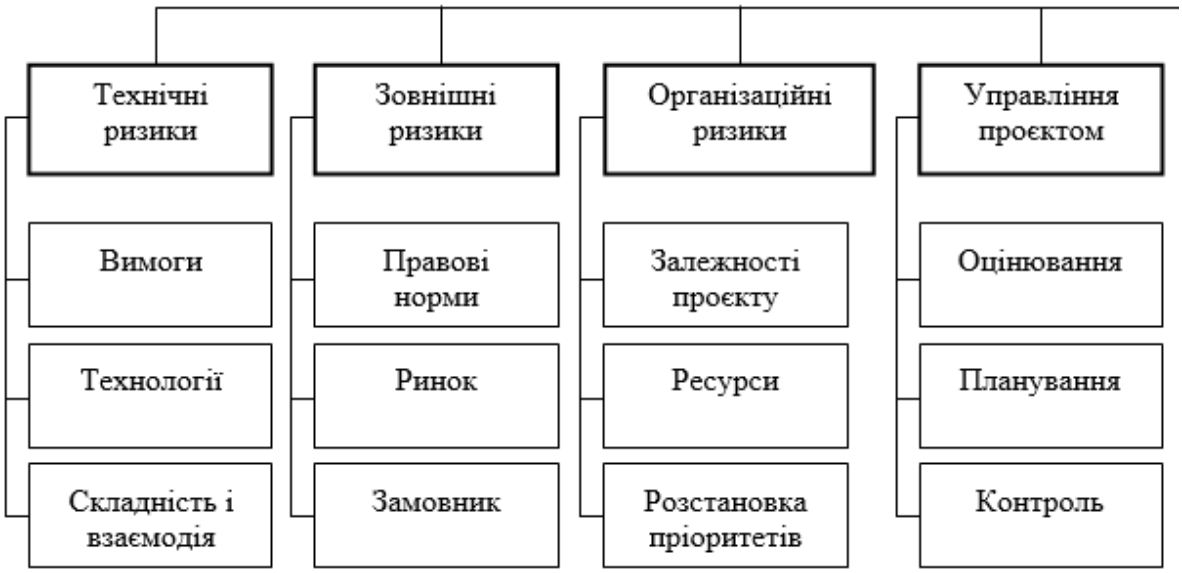


Рисунок 2.16 – Приклад ІС Ризиків

Ідентифікація ризиків – процес визначення переліку ризиків, які можуть впливати на проєкт, і документування їх характеристик. Ключова вигода даного процесу полягає в документуванні існуючих ризиків, а також в знаннях і можливостях, які це надає команді проєкту для того, щоб передбачити можливі події [1].

На вході процесу ідентифікації ризиків передбачається мати максимальне число джерел даних [1]:

- план управління ризиками;
- план управління вартістю;
- план управління розкладом;
- план управління якістю;
- план управління людськими ресурсами;
- базовий план за змістом;
- оцінювання вартості операцій;
- оцінювання тривалості операцій;
- реєстр зацікавлених осіб;
- документи проєкту;
- закупівельна документація;
- фактори середовища підприємства;
- активи процесів організації.

На виході потрібно отримати реєстр ризиків (пронумерований список ризиків з описом).

На цьому етапі ризики мають бути чітко розмежовані і про ризики мають бути точно записані такі дані:

- найменування і дата ідентифікації ризику;
- опис ризику.

При виконанні проєкту інформація про ризики має оновлюватися і доповнюватися такими даними:

- особа, відповідальна за управління ризиком;
- посилання на ІСР, де можуть виникнути додаткові роботи;
- ймовірність виникнення ризику;
- наслідки ризику;
- стратегія реагування на ризик тощо.

Ідентифікація ризиків – це процес, що вимагає інтелектуальних зусиль, абстрагування, досвіду і знань.

Можливі інструменти і методи [1]:

- огляд документації;
- методи збирання інформації:
  1) мозковий штурм;
  2) метод Дельфі;
  3) проведення інтерв'ю;
  4) аналіз першопричини;
- аналіз за допомогою контрольного списку;
- аналіз припущень;
- методи діаграм:
  1) діаграми причинно-наслідкових зв'язків;
  2) блок-схеми процесу і системи;
  3) діаграми впливу;
- аналіз SWOT;
- експертне оцінювання.

Ви можете виконати структурований аналіз проєктної документації, зокрема планів, припущень, архівів попередніх проєктів, договорів та іншої інформації. Якість планів, а також відповідність планів вимогам і припущенням проєкту можуть служити індикаторами ризиків у проєкті.

Мета **мозкового штурму** – створити вичерпний перелік ризиків проєкту. Як правило, мозковий штурм проводиться командою проєкту, часто за участю окремих експертів з різних галузей, які не входять до складу команди. Генерація ідей, пов'язаних з ризиками проєкту, відбувається під керівництвом модератора або в традиційній вільній формі мозкового штурму, або за допомогою структурованих методів проведення масових інтерв'ю. За основу може бути використана система категорій ризику, наприклад, ієрархічна структура ризику. Потім ризики підлягають ідентифікації та категоризації за видами, а їх визначення – уточненню [1].

Метод Delphi – це спосіб досягнення консенсусу серед експертів. Цей метод передбачає, що експерти з проєктних ризиків беруть участь у ньому анонімно. За допомогою анкети модератор збирає думки про важливі ризики проєкту. Відповіді узагальнюються, а потім повертаються експертам для подальших коментарів. Консенсусу можна досягти за кілька циклів цього процесу. Метод Delphi допомагає зменшити упередження при оцінюванні даних і усуває надмірний вплив індивідів на кінцевий результат [1].

Проведення інтерв'ю серед досвідчених учасників проєкту, зацікавлених сторін або експертів предметної області сприяє ідентифікації ризиків.

Аналіз першопричини – це особливий метод виявлення проблеми, виявлення першопричин, що призвели до неї, та розробки профілактичних заходів.

Контрольні списки ідентифікації ризиків розробляються на основі історичної інформації і знань, отриманих під час реалізації попередніх подібних проєктів або з інших джерел інформації. Як контрольний список ризиків можна також використовувати найнижчий рівень RBS. Хоча контрольний список може бути коротким і простим, неможливо створити вичерпний список, і тому потрібно впевнитися, що контрольний список не використовується з метою уникнути зусиль з належної ідентифікації ризиків. Команда має також приділяти увагу питанням, які не знайшли свого відображення в контрольному списку. Крім того, контрольний список потрібно час від часу скорочувати для видалення або архівування пов'язаних пунктів. Після завершення проєкту контрольний потрібно переглядати, щоб врахувати в ньому результати і поліпшити його для використання в майбутніх проєктах [1].

Аналіз припущень перевіряє обґрунтованість припущень проєкту. Цей аналіз дозволяє виявити ризики проєкту, які виникають через неточність, нестабільність, невідповідність або неповноту припущень.

Діаграми причинно-наслідкових зв'язків також відомі як діаграми Ісікави або діаграми «риб'ячий скелет», використовуються для визначення причин виникнення ризиків.

Блок-схеми процесу або системи – це своєрідне графічне зображення, яке демонструє порядок взаємодії різних елементів системи один з одним і їх причинно-наслідкові зв'язки.

Діаграми впливу – графічне подання ситуацій, що відображає причинно-наслідкові зв'язки, послідовність подій у часі та інші відношення між змінними і результатами.

Аналіз SWOT дозволяє провести аналіз проєкту з точки зору кожного з аспектів: сильних і слабких сторін, сприятливих можливостей і загроз (strengths, weaknesses, opportunities, and threats, SWOT), що робить ідентифікацію ризиків більш повною, враховуючи ризики всередині проєкту. При використанні даного методу починають з визначення сильних і слабких сторін організації, приділяючи особливу увагу або проєкту, або організації, або галузі бізнесу в цілому. Потім в процесі аналізу SWOT ідентифікують будь-які сприятливі можливості проєкту, обумовлені сильними сторонами організації, а також будь-які загрози, що з'являються внаслідок її слабких сторін. За допомогою даного аналізу також досліджують, наскільки сильні сторони організації компенсують загрози, і ідентифікують сприятливі можливості, які можна використовувати для подолання слабких сторін (рис. 2.17) [1].

	Позитивні	Негативні
Внутрішні фактори	*Сильні сторони (**S**trengths)* 1. Наявність власного фінансування 2. … 3. …	*Слабкі сторони (**W**eaknesses)* 1. Немає досвіду реалізації подібних проєктів 2. … 3. …
Зовнішні фактори	*Можливості (**O**pportunities)* 1. Можна використовувати дешеву робочу силу 2. … 3. …	*Загрози (**T**hreats)* 1. Можуть прийняти закон, що обмежує використання іноземних співробітників 2. … 3. …

Рисунок 2.17 – Аналіз SWOT

Ризики можуть бути ідентифіковані безпосередньо експертами, які мають відповідний досвід роботи в подібних проєктах або сферах бізнесу. Таких експертів має визначати керівник проєкту і запрошувати для розгляду всіх аспектів проєкту та ідентифікації можливих ризиків на основі свого попереднього досвіду і компетенції. Під час даного процесу потрібно враховувати необ'єктивність експертів.

Після того, як всі ризики ідентифіковані, проводять якісний аналіз ризиків з метою впорядкувати ризики за рівнями їх значимості.

Якісний аналіз ризиків – це процес визначення пріоритетів ризиків для подальшого аналізу або дій шляхом оцінювання й порівняння їх впливу та ймовірності виникнення. Ключова вигода даного процесу полягає в тому, що він дозволяє керівникам проєктів зменшувати рівень невизначеності і фокусуватися на високопріоритетних ризиках. Значимість ризику визначається співвідношенням двох факторів – ймовірністю ризику і наслідками ризику для цілей проєкту. При якісному аналізі ризиків ці два фактори описуються «оціночно», наприклад, ймовірність низька, середня, висока, наслідки незначні, помірні, значні тощо. Тому таке ранжування ризиків за рівнями важливості не вимагає великих тимчасових і грошових витрат, великого обсягу докладної інформації [1].

На вході процесу якісного аналізу ризиків маємо:

- план управління ризиками;
- базовий план за змістом;
- реєстр ризиків;
- фактори середовища підприємства;
- активи процесів організації.

На виході маємо оновлення документів проєкту: ранжування ризиків за рівнями важливості, перелік ризиків, які потребують додаткового аналізу, тренд результатів при повторенні якісного аналізу, тобто тенденції зміни ризиків [6–9].

Основним методом якісного аналізу ризиків є використання матриці ймовірності і впливу. Вона містить дві шкали:

– шкалу ймовірності, яка зазвичай має лінійний діапазон значень [0,1], [1,10] або [1,100];

– шкалу наслідків, яка може бути як лінійною, так і нелінійною, і відображає значимість наслідків.

Кожен ризик орієнтовно оцінюється за ймовірністю і наслідками і, згідно з матрицею, отримує певний ранг (рейтинг) важливості. Залежно від рангу, тобто клітини матриці, куди він потрапляє, ризики поділяють на низькі, середні і високі. Пороги для такого поділу ризиків в кожній організації встановлюються самостійно залежно від толерантності до ризиків [1].

Крім того, в деяких організаціях можуть складатися окремі матриці ймовірності і наслідків для окремих цілей проєкту (вартості, строків, змісту, якості) з різною шкалою і порогами [1].

Оскільки точне оцінювання ймовірності і наслідків на цьому етапі може бути ускладнене, то можна запропонувати варіант матриці ймовірності і наслідків зі спрощеною шкалою.

Кількісний аналіз ризиків зазвичай проводиться після якісного аналізу ризиків. Кількісний аналіз ризиків – процес чисельного аналізу впливу виявлених ризиків на цілі проєкту в цілому. Ключова вигода даного процесу полягає в тому, що він надає кількісну інформацію про ризики в підтримку процесу прийняття рішень з метою зменшення невизначеності проєкту.

Цей процес полягає в [1]:

– кількісному оцінюванні ймовірностей і наслідків кожного ризику, сортуванні ризиків за пріоритетами;

– визначенні ризиків, які потребують реагування та зосередження зусиль;

– кількісному визначенні величини резервів за вартістю і термінами;

– визначенні найбільш реальних сценаріїв досягнення цілей за вартістю, терміном та змістом для проєкту в цілому.

На вході процесу кількісного аналізу ризиків маємо, як мінімум, таку інформацію:

– план управління ризиками;
– план управління вартістю;
– план управління розкладом;
– реєстр ризиків;
– фактори середовища підприємства;
– активи процесів організації.

На виході отримуємо поновлення документів проєкту:

– перелік ризиків за пріоритетами, оцінка наслідків ризиків;
– імовірнісний аналіз проєкту;
– тренди результатів (при повторному аналізі).

Кількісний аналіз ризиків проводиться щодо тих ризиків, які в результаті процесу якісного аналізу ризиків були класифіковані як потенційні та істотним чином впливаючі на конкурувальні вимоги проєкту. У процесі кількісного аналізу ризиків оцінюється вплив цих ризиків на цілі проєкту. Він використовується, в основному, для оцінювання спільного впливу всіх ризиків на проєкт. Коли ризики потрапляють в кількісний аналіз, даний процес може використовуватися для присвоєння числового рейтингу пріоритетності цих ризиків окремо [1].

У деяких випадках виконання процесу кількісного аналізу ризиків неможливо в зв'язку з відсутністю необхідних даних для розробки відповідних моделей. Керівник проєкту має користуватися експертною оцінкою для визначення необхідності та доцільності кількісного аналізу ризиків. Вибір методу (методів) аналізу в кожному конкретному проєкті визначається наявністю часу і бюджетом, а також потребою в якісному і кількісному описі ризиків і їх впливів. Щоб визначити, чи був ризик проєкту успішно знижений, кількісний аналіз ризиків потрібно, за необхідності, повторно провести в рамках процесу контролю ризиків. Аналіз тенденцій може вказати на необхідність приділити більше або менше уваги відповідним діям з управління ризиками [1].

2.5 Програмні засоби для управління проєктами

Впровадження управлінських інформаційних систем в організації сьогодні перестало бути лише засобом підвищення ефективності існуючої системи управління. Постійне вдосконалення методів управління організацією, що підкріплюється використанням сучасного програмного забезпечення, є умовою успішного функціонування компанії на ринку. Розвиток інформаційних технологій постійно нагадує нам про закон переходу кількості в якість: бажане стає можливим, недоступне – доступним і економічно ефективним. Одним із завдань керівника стало крокувати в ногу з прогресом в інформаційних технологіях, щоб не відстати від конкурентів [1].

Microsoft Project наразі є найпоширенішою у світі системою УП (управління проєктами). У багатьох західних компаніях MS Project став звичайним доповненням до Microsoft Office навіть для звичайних співробітників, які використовують його для планування простих робочих графіків (рис. 2.18).

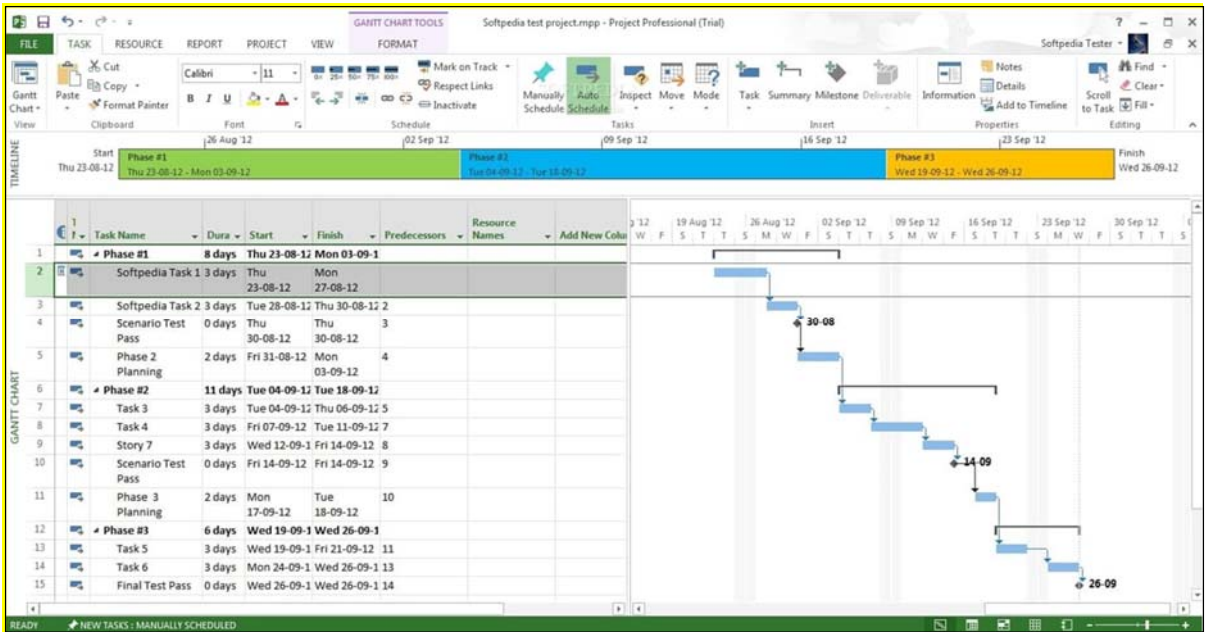


Рисунок 2.18 – Інтерфейс MS Project

Відмінною рисою MS Project є його простота. Розробники MS Project не прагнуть вносити в пакет більш складні алгоритми календарного або ресурсного планування. При цьому значна увага приділяється використанню сучасних стандартів, які дозволяють ефективно інтегрувати пакет з іншими додатками.

Підтримка Microsoft Mail і Microsoft Exchange дозволяє полегшити та організувати групову роботу з проєктами. Налаштування повідомлень для команди проєкту дає можливість визначення складу даних проєкту, які надсилаються учасникам проєкту електронною поштою, та встановлення обмежень на виправлення інформації, яка пересилається одержувачами. Зберігання проєктів у папках Exchange надає додаткові інструменти для розмежування доступу до файлів проєкту [1].

Щоб швидко залучити користувача-початківця, MS Project надає, крім звичайних інструментів, також можливість поетапної розробки проєкту (Create Your First Project і Cue Cards) та інтелектуальні підказки (Answer Wizard). На жаль, Project поки не українізували, так що для ефективного використання цих засобів необхідно знання англійської мови, зокрема специфічної термінології управління проєктами.

Серед переваг пакета також потрібно відзначити досить зручні та гнучкі засоби створення звітів. З заготовок можна вибрати основні типи звітів (Report Gallery). Можливість мати до шести планів для кожного проєкту одночасно може підвищити ефективність аналізу. У той же час MS Project надає мінімальний набір інструментів для планування та управління ресурсами. Додаткові функції проєкту також містять імпорт/експорт даних у форматах ASCII, CSV, Excel, Lotus 1-2-3, dBASE та FoxPro, інструменти запису макросів, Visual Basic [1].

MS Project можна рекомендувати для планування простих проєктів непрофесіоналам і користувачам початківцям.

Центральний програмний продукт сімейства Primavera, Primavera Project Planner, добре відомий серед професійних менеджерів проєктів у всьому світі. Project Planner використовується для управління середніми та великими проєктами в різних сферах, хоча найбільшого поширення продукт отримав у сфері управління будівельними та інженерними проєктами.

Primavera Project Planner забезпечує досить стандартний графічний інтерфейс для всіх подібних систем (рис. 2.19), але має кілька додаткових можливостей. По-перше, це вміння групувати та організовувати роботу за різними ознаками на різних рівнях деталізації проєкту, що дозволяє подати інформацію в більш зручній формі для конкретної управлінської ситуації. Наприклад, за допомогою цих інструментів усю інформацію про проєкт можна згрупувати за фазою проєкту на першому рівні ієрархії, за відповідним ресурсом – на другому та відсортувати за датою початку на третьому рівні. Кожній групі можна встановити власний шрифт і колір (текст і фон), розрив сторінки.

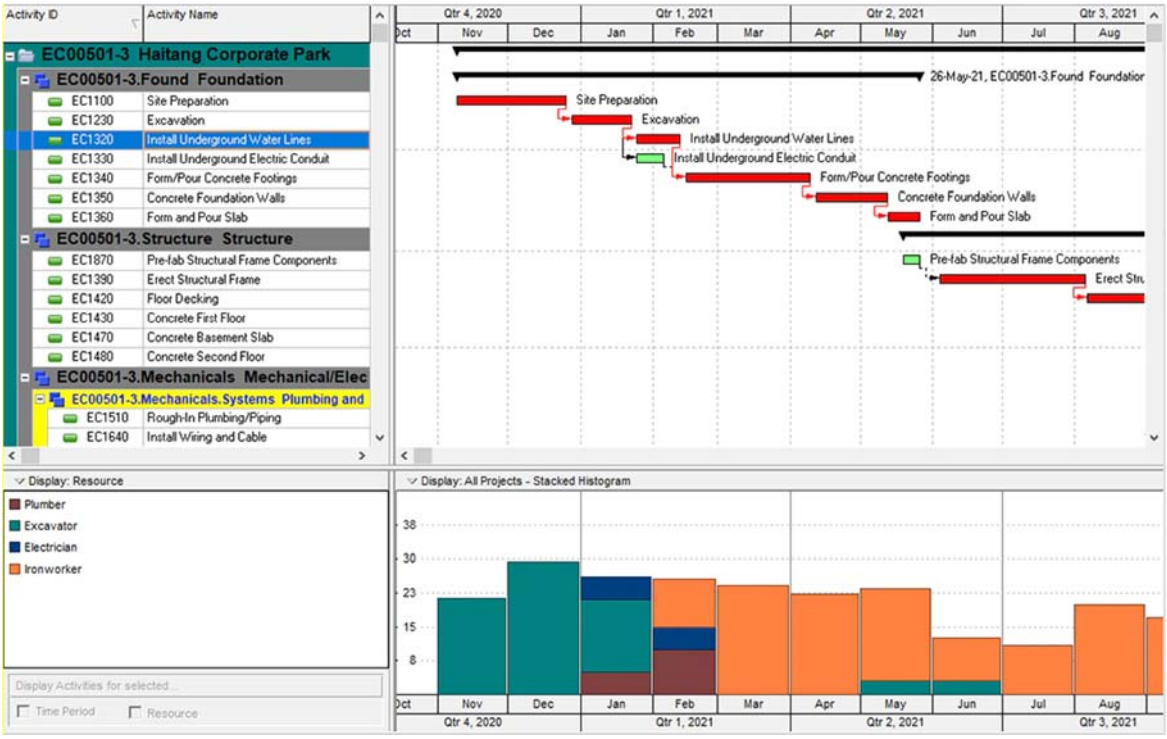


Рисунок 2.19 – Інтерфейс Primavera Project Planner

Ще одна корисна функція – можливість розділити екран по горизонталі на дві частини, кожну з яких можна переглядати окремо. Це дозволяє одночасно переглядати різні частини проєкту.

Крім того, Project Planner має деякі відмінності від інших інструментів планування ресурсів.

При описі ресурсу можна вказати нормальну та максимальну кількості доступних ресурсів, а також його ціну через шість інтервалів часу.

Ресурс може бути призначений як керований (обсяг керованого ресурсу, призначений для завдання, вплине на тривалість його виконання). Наприклад, визначивши, що працівники є керованим ресурсом, а бригадири – ні, можна скоротити час, необхідний для виконання завдання прокладання траншеї, призначивши більше працівників. Збільшення ж чисельності бригадирів не вплине на тривалість роботи [1].

При плануванні завантаження ресурсів може виникнути необхідність описати нелінійний профіль споживання ресурсів як окрему задачу. Project Planner дозволяє вам описувати різні криві розподілу ресурсів, пропонуючи дев'ять стандартних кривих і можливість визначати власний профіль споживання, розбиваючи часову фазу завдання на 10 періодів.

Засоби автоматичного перепланування завдань з урахуванням обмеженості ресурсів особливо важливі для великих проєктів, коли керівник не в змозі самостійно проаналізувати причини нестачі ресурсів і знайти рішення для кожного завдання. Project Planner дозволяє вибрати режим перерахунку розкладу та вибрати критерій перепланування робіт, що забезпечує більш короткий розклад. Режими перерахунку містять вирівнювання вперед (визначення можливої дати завершення проєкту при заданій даті початку), вирівнювання назад (визначення найпізнішої допустимої дати початку проєкту), згладжування перевантажень ресурсів у межах тимчасових резервів робіт або в межах заданого інтервалу [1].

Крім того, можливий перерозподіл призначення робіт між згрупованими ресурсами.

Інструменти для підтримки багатопроєктного середовища керування в Project Planner мають можливість визначати ієрархії та права доступу до основних проєктів і підпроєктів. Менеджер-координатор проєкту має право редагувати головний проєкт і всі підпроєкти. Менеджер підпроєкту має право додавати ресурси до словника ресурсів, але не видаляти їх або змінювати ціни. Якщо для вирішення ресурсних конфліктів всередині підпроєкту потрібні дані з іншого підпроєкту, менеджер може це зробити лише при наданні йому додаткових повноважень менеджером-координатором проєкту. Однак планування ресурсів для всього проєкту може здійснювати тільки менеджер-координатор. Тільки він може визначити зв'язки між підпроєктами. Порівняно з багатьма іншими програмними продуктами, які також дозволяють керувати кількома проєктами, відмінною рисою Project Planner є детальний опис принципів управління кількома проєктами в документації, де вони розглядаються з двох точок зору: менеджера-координатора проєкту та менеджера підпроєктів (хоча вважається, що тема багатопроєктного менеджменту потребує додаткового підручника) [1].

На додаток до Project Planner, Primavera Systems надає польову систему для УП – SureTrak. Цей програмний продукт орієнтований на невеликі проєкти, підпроєкти, роботу конкретних виконавців з фрагментами проєктів. SureTrak має ті ж інструменти з точки зору організації проєкту за допомогою фільтрації коду та інформації, встановлення лімітів і планування, але в той же час є низка обмежень і додаткових функцій.

Серед обмежень – відсутність управління кількома проєктами та фрагментація проєктів, менші розміри проєкту та більш скромні засоби звітності. Однак SureTrak має календарі ресурсів і, як наслідок, можливість обчислити тривалість роботи з урахуванням узгодження календарів. Крім того, ресурси мають додаткову категорію – дохід.

SureTrak імпортує/експортує файли у форматах Project Planner і MS Project. Таким чином, працюючи разом, Project Planner і SureTrak пропонують масштабований підхід до управління проєктами різного розміру та складності. Крім перерахованих вище продуктів сімейства Primavera, може зацікавити система аналізу ризиків проєкту Monte Carlo for Primavera [1].

Ще один відомий бренд у світі проєктного менеджменту – Artemis. Традиційно програмне забезпечення сімейства Artemis використовувалося для управління великими інженерними проєктами. Сьогодні Artemis International розповсюджує серію програм під цим брендом під загальною назвою ArtemisViews.

Сімейство ArtemisViews складається з набору модулів, які автоматизують різні аспекти управління проєктами: ProjectView, ResourceView, TrackView, CostView. Усі модулі сумісні з даними, працюють в архітектурі клієнт/сервер, підтримують стандарт ODBC і легко інтегруються з популярними СУБД Oracle, SQLBase, SQLServer, Sybase.

Кожен модуль може працювати самостійно або в поєднанні з іншим програмним забезпеченням. Ціна на це, традиційно дороге програмне забезпечення, розраховується на основі замовленої конфігурації.

ResourceView – це спеціалізована система для планування та контролю використання ресурсів як у проєктному середовищі управління, так і для поточних робіт. У системі реалізовані інструменти підтримки координації розподілу ресурсів менеджерами. Графічна панель управління ресурсами дозволяє менеджерам планувати, контролювати й оптимізувати своє навантаження шляхом перерозподілу черги робіт за наявністю ресурсів.

TrackView надає засоби для збереження фактичної інформації про обсяги виконаної роботи, контроль за станом і вартістю поточних робіт (проєктних і непроєктних). Система дозволяє інтегрувати дані для різних рівнів управління в організації (від рядових виконавців, які ведуть інформацію про свої завдання, до вищого керівництва, яке може отримати зведені дані про фактичні витрати та обсяги робіт) [1].

CostView забезпечує підтримку центрального репозитарію для інформації про всі витрати та доходи проєкту. Пакет дозволяє аналізувати економічну ефективність контрактів, будувати таблиці руху грошових коштів, прогнозувати витрати та розраховувати внутрішню норму прибутку проєктів. Звичайно, ArtemisViews дозволяє створити потужне інтегроване рішення, однак витрати, пов’язані з купівлею та впровадженням цього програмного забезпечення, значно обмежують коло потенційних користувачів.

З розвитком веб-технологій багато команд та ІТ-компаній обирають веб-сервіси для управління своїми продуктами. Ось список найвідоміших з них:

– JiRA;
– Redmine;
– Trelllo;
– Monday;
– Todoist.

Також, в процесі реалізації проєкту важливо використовувати сучасні глобальні інформаційні веб-ресурси:

- соціальні мережі;
- меседжери;
- геопортали (веб-ГІС);
- тематичні ресурси.

Дані ресурси можуть бути використані для:

- пошуку інформації про стейкхолдерів проєкту та налагодження первинної комунікації з ними;
- пошук конкурентів та детальне визначення їх сильних і слабких сторін;
- аналіз ринку;
- визначення цільових груп;
- пошук кандидатур з відповідною кваліфікацією при формуванні команди тощо.

Для розв'язання задач просторового аналізу, оптимізації логістики, просторової візуалізації можуть бути використані сучасні ГІС [1]:

- ArcGIS (зокрема веб-сервіс ArcGIS online);
- MapInfo;
- QGIS;
- Google Maps;
- OpenStreetMap;
- Carto.

2.6 Управління комунікаціями в ІТ-проєкті

Комунікації в проєкті – це ефективний обмін інформацією між учасниками проєкту.

Управління комунікаціями проєкту – розділ управління проєктами, що містить завдання і процедури, які необхідні для забезпечення інформаційних потреб учасників проєкту.

Комунікації (лат. Communicato – повідомлення, передача) – смисловий та ідеально-змістовний аспект соціальної взаємодії.

Основна функція комунікацій – досягнення соціальної спільності при збереженні індивідуальності кожного її елемента.

Структура комунікації:

1) двоє учасників-комунікантів, наділених свідомістю і які знають норми деякої семіотичної системи, наприклад, мови;

2) ситуація (або ситуації), яку вони прагнуть осмислити і зрозуміти;

3) тексти, що виражають сенс ситуації в мові або елементах даної семіотичної системи;

4) мотиви і цілі, які роблять тексти спрямованими, тобто те, що спонукає суб'єктів звертатися один до одного;

5) процес матеріальної передачі текстів.

Комунікації можуть бути класифіковані за цілою низкою ознак.

За типом відносин між учасниками:

– міжособистісна;
– публічна;
– масова.

За типом використовуваних засобів:

– мовна;
– паралінгвістична (жест, міміка, мелодія);
– матеріально-знакова (текст).

Стосовно проєкту:

– внутрішня (команда проєкту);
– зовнішня (ЗМІ).

Щодо відносин між учасниками:

– формальна (звіт, наказ);
– неформальна (переговори, дискусії, e-mail).

За ієрархічністю структури:

– вертикальна (звіт, наказ);
– горизонтальна (збирання та передача вихідних даних).

За взаємодією між учасниками:

– офіційна (річний звіт, розкриття інформації);
– неофіційна (листування).

За способом передачі:

– письмова;
– усна.

За каналами передачі і сприйняття:

– вербальна (мова);
– невербальна (поза, міміка, жести).

За відкритістю (доступністю):

– конфіденційна;
– така, що не містить комерційну таємницю.

Говорячи про комунікації окрему увагу потрібно приділити нарадам.

Порядок роботи нарад:

– підготовка (графік, порядок, збирання та підготовка матеріалів, презентації, чек-лист, приміщення та обладнання);

– проведення (ролі, протокол, прийняття рішень, терміни, відповідальний);

– виконання рішень (протокол, матеріали, звіти про виконання);

– контроль виконань рішень (терміни, відповідальний, форма контролю).

Якщо графічно зобразити частоту проведення нарад протягом життєвого циклу проєкту, то отримаємо графік, зображений на рисунку 2.20.

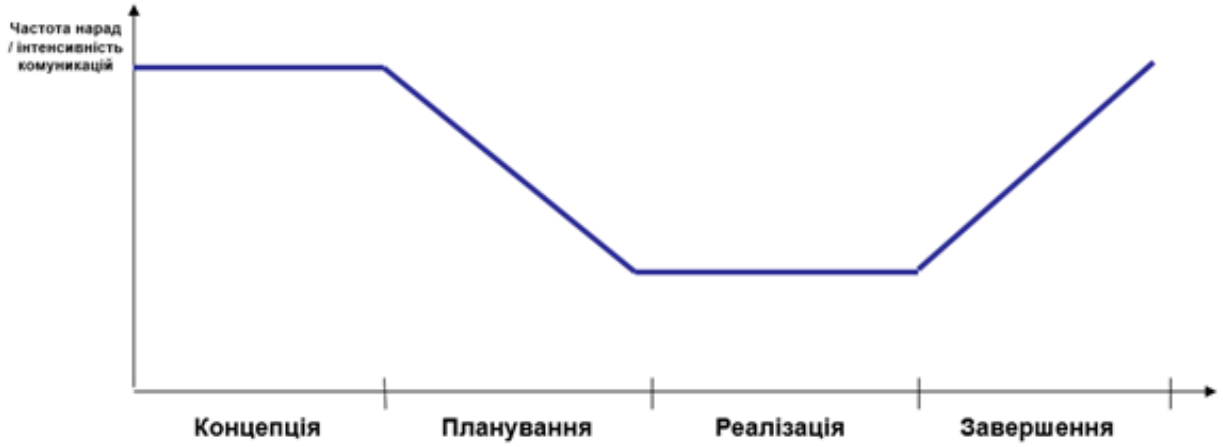


Рисунок 2.20 – Інтенсивність нарад протягом життєвого циклу проєкту

В ІТ-проєктах дуже поширеними є віртуальні наради. При їх плануванні і проведенні потрібно дотримуватись таких правил:

– запрошення на нараду надсилається всім учасникам, відповідь про прийняття або відхилення запрошення висилається організатору через електронну пошту;

– всі учасники зобов’язані приєднуватись до наради завчасно. Підключення до Інтернету, налаштування і тестові дзвінки здійснюються учасниками заздалегідь, до початку наради;

– під час наради необхідно строго дотримуватися порядку;

– будьте професійні, ввічливі, слухайте, не перебиваючи, говоріть по черзі;

– організатор наради (meeting organizer) відповідає за дотримання часових рамок і регламенту, і тільки він має право перебити і зупинити учасника, щоб передати слово наступному;

– для кожної наради організатор готує порядок, протокол і забезпечує розсилання всім учасникам;

– після отримання протоколу всі учасники зобов’язані його погодити, прийняти або внести свої пропозиції, доповнення та зауваження і вислати організатору, після чого організатор зобов’язаний вислати узгоджений протокол учасникам для виконання;

– рішення узгодженого протоколу є обов’язковими до виконання всіма учасниками наради.

Також окрему увагу потрібно приділити різним формам інформаційних носіїв і їх характеристикам. Порівняльна таблиця різних форм наведена нижче (табл. 2.1).

Таблиця 2.1 – Порівняння інформаційних носіїв

Форма інформаційного носія	Надійність	Точність	Рівень сотворень	«Людський фактор»
Усна	Низька	Низька	Високий	Високий
Письмова	Висока	Середня	Середній	Середній
Електронна	Середня	Висока	Низький	Низький

Залежно від форми здійснення комунікацій прийнято розрізняти три основні моделі документообігу:

- консервативна модель;
- сучасна модель;
- неділова.

Як в цих моделях розподіляється форма здійснення комунікації наведено в табл. 2.2.

Таблиця 2.2 – Основні моделі документообігу

Форма здійснення комунікацій	Консервативна модель	Сучасна модель	Неділова
Усна	15%	15%	65%
Письмова	65%	20%	20%
Електронна	20%	65%	15%

На даний час варто використовувати сучасні меседжери та соціальні мережі для організації електронної форми здійснення комунікацій під час виконання ІТ-проєкту. Дані ресурси прийнятні для ведення як ділової комунікації, так і неформального спілкування між учасниками команди.

2.7 Основи Agile

Розглянемо для початку водоспадну модель. Лікар Ройс створив так звану водоспадну модель розробки програмних продуктів. Вона швидко завоювала популярність на Заході, і якийсь час тому за цією моделлю працювала переважна більшість компаній-розробників. Що вона є?

Розробка продукту проходить низку етапів:

- збирання вимог;
- їх аналіз;
- створення архітектури;
- створення дизайну системи;
- кодування;
- тестування;
- викладення;
- експлуатація.

В ідеальному світі ми пройшли б цими рівнями зверху вниз, як тече водоспад, і в кінці у нас вийшла б хороша система. У чому полягало рішення доктора Ройса? Він запропонував писати багато документації, обробляти ризики, повторювати кілька разів якісь етапи. У результаті вийшла великовагова водоспадна модель. Вся промисловість комерційного софту сформувалася 1990-х роках. І якщо в Європі та США існували «тяжкі» методології, то у нас не було жодних. Збиралася група людей, які просто намагалися зробити добрий софт. Обидві проблеми – відсутність методології у нас і великовагові схеми на Заході – добре вирішує Agile.

Отже, навіщо застосовується методологія Agile?

Прискорити вилучення товарів з ринку. Якщо ви хочете зробити щось швидше, ви мєте робити це відповідно до Agile. Дуже простий приклад. Є дві компанії, у них приблизно однаковий бізнес. Пишуть технічні характеристики, потім проєктують систему і рисують дизайн – це модель водоспаду, розробка якої може зайняти кілька місяців. Друга компанія, що працює за Agile, в цей же час може вже мати запущений веб-сайт, випущене програмне забезпечення, заробіток та захоплення ринку, що найважливіше.

Керування змінами пріоритетів. Це, мабуть, дуже болісна проблема практично для всіх компаній. Якщо ви розробляєте проєкт, який триває хоча б кілька місяців, то обов'язково зміняться вимоги. Звичайно, якщо це не софт, наприклад, для супутника чи марсоходу. Хоча навіть супутникам та марсоходам зазвичай заливають свіжу версію софту, коли вони прилітають у точку призначення. Коли справа доходить до комерційної розробки, проблема в тому, що програмісти, аналітики та дизайнери, ніколи не знають, що потрібно не тільки замовнику, який нам платить, а й користувачам. Зазвичай всі підходять до питання так: поки користувач не спробує функціонал сайту або програми, ви не знаєте, потрібне вам це чи ні.

Покращення взаємодії між IT та бізнесом. Це головний біль, особливо для великих компаній, адже бізнес періодично змінює вимоги, кожен говорить своєю мовою. В результаті сторони не розуміють одна одну.

Відповіддю на всі ці виклики став Маніфест гнучкої розробки програмного забезпечення. Він складається з кількох частин. Перша частина називається «Цінності» (Values). Це чотири «зважування»:

1. Якщо ви хочете побудувати гнучкий процес, вам потрібно взаємодіяти та спілкуватися один з одним. У чому це виявляється – розглянемо нижче на прикладі Scrum. При цьому ви можете (і обов'язково будете) використовувати якісь інструменти та процеси, наприклад, трекери – JIRA, Redmine тощо. Але ваша робота має спиратися на різні мітинги, зустрічі та взаємодію, а не на налаштування трекерів або TFS (якщо казати про Microsoft стек).

2. Продукт, який ми виготовляємо, набагато важливіший за документацію, яку ми виготовляємо. Вище наведено приклад з двома компаніями: одна має готовий продукт, який можна дати користувачам, замовнику, за-

хоплюючи ринок; а інша пише технічні характеристики, рисує моделі тощо. Вся ця документація, якою користувач не може скористатися через непідготовленість продукту, не приносить користувачеві ніякої цінності. Якщо ми навчимося працювати, зводячи ці кроки до мінімуму або роблячи їх невеликими, ми матимемо більш гнучкий процес.

3. Співпраця та взаємодія з замовником важливіші за жорсткі договірні обмеження. Зазвичай укладається договір про те, що розробник зобов'язується виконати зазначений обсяг робіт на конкретну дату за певну суму. Звичайно, технічне завдання (ТЗ) додається до договору. Тобто, фіксуються час, обсяг робіт та терміни виконання. Це називається фіксованою ціною. Такий підхід не дуже хороший, якщо ви хочете довгостроково працювати і бути гнучким. І тут правильніше будувати партнерські відносини з замовником. Якщо говорити про підрядне проєктування, то воно, зазвичай, виливається в договори за схемою «час – матеріали», коли розробнику просто оплачують витрачений час. Найголовніше, що саме з цього починається пошук партнерства і ситуації Win-Win, коли виграють і замовник, і його підрядник.

4. Готовність до змін у зважуванні з дотриманням початкового плану.

У Agile є план, оцінки та прогнози. Але якщо у вас є якийсь первісний план для річного проєкту, а ви через три місяці вже надали якусь версію продукту, користувачі його спробували, ви зняли метрики, подивилися, що і як вони використовують, дізналися щось нове, то після цього первісний план можна майже повністю змінити.

Наведемо новий приклад. Припустимо, що з'явилась маленька фіча на сайті з пошуку роботи, коли ваші навички може підтверджувати будь-хто – поставив вам плюсик, і з'явиться напис, що стільки-то людей підтвердили ваші навики. Це спеціально запустили у дуже простому вигляді, щоб подивитися, як до цього поставляться користувачі. Розробники виходили з логіки, що буде взаємодія, коли користувачі один одного плюсують. Зняли статистику, і виявилося, що ці плюси ставлять, мабуть, після співбесід HR (human resources). Тобто далі можна розвивати цю фічу у бік HR, або якось її модифікувати, щоб вона була корисніша для претендентів. Таким чином, у Agile існує чотири цінності:

– люди та взаємодія між ними;

– робочий продукт;

– співробітництво та налагодження партнерських відносин із замовником;

– готовність до змін.

Цінності спричиняють 12 принципів Agile:

1. Найвищою цінністю є задоволення потреб замовника шляхом регулярної та ранньої доставки цінного програмного забезпечення. Якщо замовник хоче отримати від нас великого слона, але ми можемо віддати йому частину цього слона не через рік, а через три місяці, потім через три місяці

ще частину, а потім щомісяця видавати шматочки, то чим частіше ми це робитимемо і чим раніше, тим краще.

2. Готовність завжди змінити вимоги, навіть на пізніх стадіях проєкту, якщо дізнаємося щось нове. Таким чином створюється конкурентна перевага для бізнесу або зовнішнього клієнта. Наприклад, є дві компанії: одна написала технічні характеристики і виготовила продукт за рік, а друга зробила концепцію продукту (неважливо в якому вигляді) і поступово його розгортає і розгортає. Тоді цей продукт краще відповідатиме вимогам клієнтів, користувачів і ринку в цілому.

3. При використанні Agile робочий продукт випускається якомога частіше. Маніфест передбачає терміни – від кількох тижнів до кількох місяців. Фактично, це тиждень/місяць, якщо ви використовуєте Scrum. А якщо ви робите веб-проєкт, тоді ви, зазвичай, використовуєте один із варіантів Kanban, тому випуски можна робити щодня. Зазвичай серйозні проєкти випускають кілька випусків щодня, що є великою проблемою для конкурентів. Можливо, це редагування помилок, але вони завжди знають, як додати щось цінне для їх користувачів.

4. Бізнес обов'язково має працювати разом із програмістами, допомагати їм зрозуміти специфіку даного ринку. Наприклад, програмісти сайту для пошуку роботи мають розуміти, як працює HR, і як люди шукають роботу. Якщо ви працюєте в банку, то вам потрібне розуміння принципу роботи банку в цілому і дуже докладні відомості про галузь, за яку ви відповідаєте. Найчастішою проблемою є недоступність бізнесу – коли розробник не може отримати у співробітника потрібну інформацію. При використанні Agile важливо уникати виникнення подібних ситуацій.

5. Команда – один з наріжних каменів Agile. Найкращих результатів досягає команда вмотивованих фахівців. Є геніальне зауваження, що ефективність Scrum залежить від керівника. У Agile керівник, перш за все, має створювати умови для команди та забезпечувати всебічну підтримку, проводити коучинг, стежити за атмосферою у колективі.

6. Є багато досліджень, які показують, що найкраще спілкування віч-на-віч. Причому бажано, щоб був якийсь засіб візуалізації, на якому можна писати: аркуш паперу, дошка зі стікерами тощо. Найпростіший і найефективніший спосіб дізнатися про вимоги клієнта, замовника чи користувача – поговорити з ними.

7. Діючий продукт. Про нього повторюватись не будемо. Ступінь готовності проєкту має вимірюватися не словами про те, що ТЗ вже написано та 50% макетів нарисовано, а кількістю функціоналу, випущеного у production.

8. У Agile важливий ритм, постійні вдосконалення. Бізнес і програмісти мають завжди вміти робити процес стійким, постійно його вдосконалювати.

9. Менеджери зазвичай не люблять говорити про це з розробниками, але Agile взагалі не працюватиме, якщо ви написали «поганий» код. У вас

має бути хороша гнучка архітектура, до якої можна додавати різні елементи та легко змінювати їх у разі потреби. І якщо команда не приділить максимум уваги технічній якості (писати хороший код, використовувати інженерні методи, автоматизувати процеси), то у вас не буде ніякого Agile.

10. Простота. Це проявляється в технічній складовій, в дизайні. Це один із принципів екстремального програмування. Простота також дуже важлива з точки зору випуску продукту: коли хочеш «розрізати» того «слона», краще почати з простої частини.

11. Менеджер (керівник, Scrum-коуч, Agile-коуч) змінює свою роль у команді: він не стільки займається організацією процесу, скільки навчає команду, тому команда має бути самоорганізованою. Існують спеціальні стратегії, як зробити групу людей самоорганізованою командою.

12. Команда має постійно аналізувати свою роботу, процеси: що сталося, як вони цього досягли, і постійно вдосконалювати організацію роботи.

Як результат можна сказати, що Agile – це серія підходів до розробки програмних продуктів шляхом безперервного та швидкого постачання цінного робочого функціоналу самоорганізованою командою професіоналів у співпраці з замовником.

Отже, у нас виходить піраміда, що складається з чотирьох цінностей, на яких побудовано 12 принципів. Наразі з’являються конкретні практики.

Одна з цінностей Agile свідчить: ми маємо вибудовувати робочий процес, налагоджуючи взаємодію та комунікації між людьми. Це виливається у практичні кроки. Наприклад, у ранкові стендапи, коли команда усно синхронізує свою діяльність наступного дня. Можна замість стендапів кожному писати звіт про зроблене вчора, але це вже не буде Agile, тому що виникає потік малозначущої документації. Розглянемо практики, які найчастіше використовуються у компаніях, що практикують гнучку розробку:

1. На першому місці стендапи. Навіть якщо компанія повністю зафіксувала використання та адаптацію Agile, зазвичай все одно залишаються стендапи. Якщо вам не вдається їх використовувати, значить, у вас зовсім не організована команда. Практика проста: кожен день у певний час команда збирається та синхронізує свою діяльність.

2. На другому місці планування ітерацій, коли команда планує обсяг робіт на найближчу ітерацію. Це питання про те, що в Agile немає планування. Якщо ітерація триває два тижні, то треба намагатися зробити план, декомпозицію, можливо, розбити бізнес-завдання на технічні завдання.

3. Unit-тестування – одна з найпростіших інженерних практик, яку можна швидко почати використовувати. За наявності проблем з якістю, близько 60–70% багів можна відловити unit-тестуванням.

4. Планування релізів. Тут потрібно вже планувати великими шматками – що коли випускається.

На рисунку 2.21 зображено, як можна графічно уявити ітеративність виконання робіт.

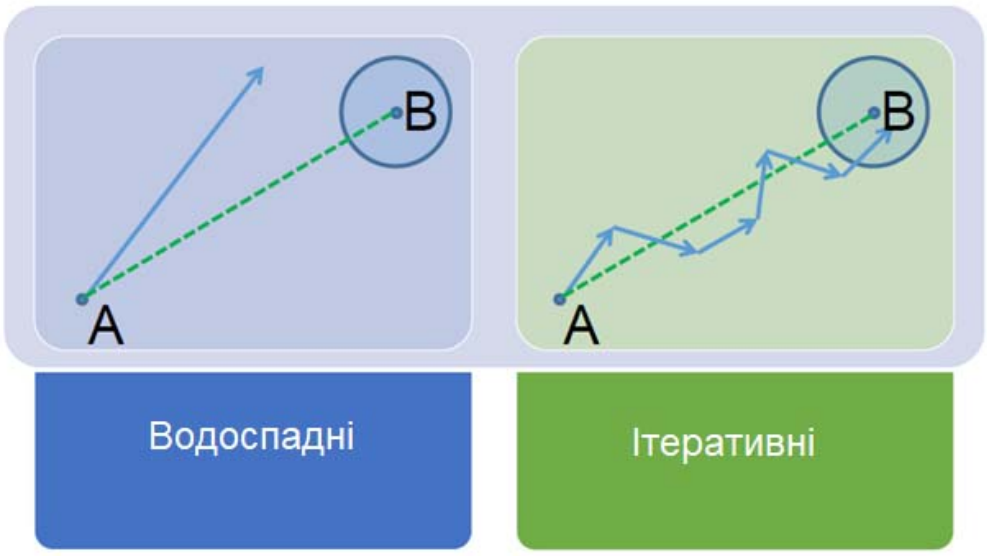


Рисунок 2.21 – Графічне подання ітеративності виконання робіт

Нам потрібно перейти від точки А до точки Б. «Перейти» означає «отримати зворотний зв'язок». Скажімо, у нас є GPS, ми можемо дістатися до точки та перевірити, чи правильно ми дійшли. Модель водоспаду – це, насправді, один великий крок. Тобто, ми кудись прийшли, випустили продукт на ринок, і тільки тоді можемо зрозуміти, що саме у нас вийшло. Ітеративна модель – це серія кроків. Робимо перший крок, знімаємо метрики, які використовуються в продукті, а потім коригуємо наступні кроки. Це дозволить якщо не підійти до точки Б, то точно опинитись в її околицях.

У комерційному розвитку постійно змінюються умови: видаються нові закони, змінюються вимоги бізнесу, конкуренти раптом випускають те, що потрібно зробити краще за них. В результаті нам потрібно потрапити з точки А не в точку Б, а в точку В. Ітеративна модель передбачає, що після випуску першого релізу ми знімаємо метрики, щось переробляємо, робимо наступний реліз і так далі. І в якийсь момент ми розуміємо, що конкурент випустив якусь функцію, і нам потрібно піти не так. На цьому моменті ми можемо зупинитися, прийняти інші вимоги і почати рухатися в пункт В. Завдяки високій мінливості умов у процесі створення продукту ви постійно дізнаєтеся про щось нове. І це одна з переваг ітераційної моделі.

Ще важливий момент: коли менеджери (і навіть розробники) не дуже глибоко занурені в технічну частину, їм здається, що можна зробити програму ітеративно, зібрати по шматочках, як пазл. Це помилка. Розробник має уявляти повністю і в деталях кожен елемент картини, і почати його робити за п'ять кроків. При цьому потрібно пам'ятати, що умови можуть змінитися. Це називається інкрементальним підходом («інкремент» – додавання чогось).

Щоб працювати ітеративно та інкрементально, потрібно діяти трохи інакше. У розробника у голові має бути якась концепція, яку він поступово опрацьовує.

Якщо ви працюєте з «водоспадом», то, зазвичай, у вас фіксований зміст проєкту. В Agile це відбувається інакше: є команда та фіксовані відрізки часу (я говорю про Scrum), наприклад, двотижневі ітерації. Виходячи з цього планується обсяг робіт, який може виконатись за цей час.

Якщо взяти класичний підхід і відповісти на запитання успішний цей проєкт чи ні, то ми маємо його зробити вчасно, не перевищивши бюджет. Якщо дивитись з позиції Agile, то проєкт тим успішніший, чим більше ми поставили цінності замовнику.

Хенрік Кніберг має таку метафору – Agile-парасольку. Це ті методології, які є гнучкими. Найбільша – Scrum, екстремальне програмування (XP), DSDM, Crystal, FDD, Kanban. Більше половини компаній, які використовують Agile, використовують Scrum. На другому місці комбінація Scrum та XP, коли беруться управлінські практики зі Scrum та додаються інженерні. Окремо зазначимо, що комбінація Scrum та Kanban використовується приблизно у 8% компаній. Зараз це один із трендів, мода. Назва Scrum надійшла з регбі і перекладається як «битва». Простіше кажучи, при виникненні спірної ситуації команди вишиковуються, вкидається м'яч, і потрібно один одного перештовхати. До розробки продукції цей термін вперше застосували в 1980-х роках два японці – Хіротака Такеуті та Ікудзіро Нонака. Це хороші дослідники у галузі менеджменту. Зокрема, вони написали документ «The New New Product Development Game». Тут вживання слова «new» 2 рази не є помилкою. Просто назва перекладається як «Нова гра для розробки нових продуктів». Що зробили ці два японці? Вони проаналізували, як різні компанії виробляють свої продукти. Причому навіть не програмне забезпечення, а різноманітну техніку та електроніку. Автори розділили виявлені підходи на три типи (рис. 2.22).

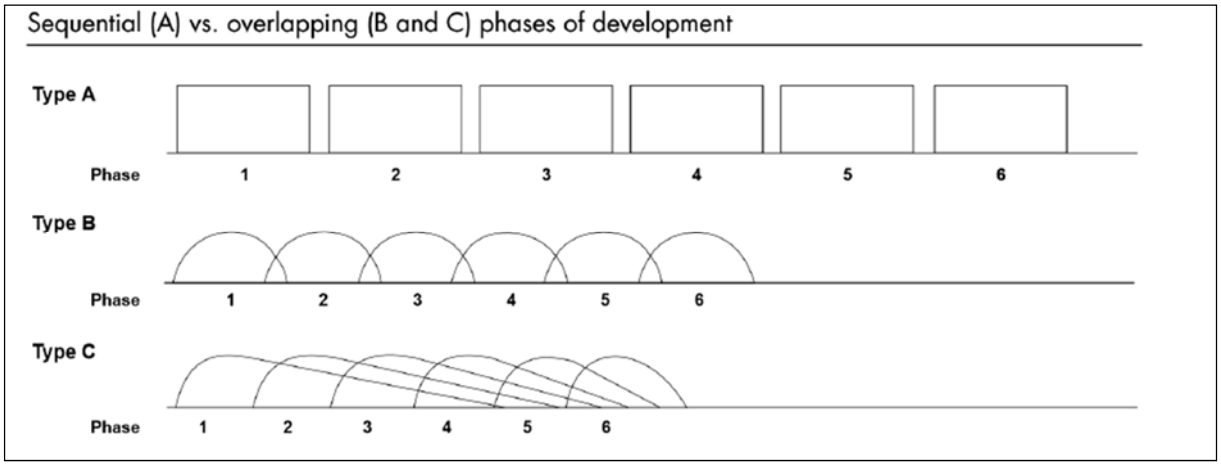


Рисунок 2.22 – Типи продуктів за методологією Scrum

Перший тип (A): існує фаза розробки, все жорстко і послідовно.

Другий тип (B): зв'язки перетинаються, можна отримати зворотний зв'язок. Програміст запрограмував щось, програмує ще, віддає тестуваль-

нику, він дає свої коментарі, програміст встигає їх обробити до завершення своєї роботи.

Третій тип (C): кожна фаза перетинається більш ніж однією фазою. Програміст програмує ще щось, а його перші завдання тестуються, викладаються в production, з них знімаються метрики, і програміст може внести зміни.

Тип С двоє японців порівнювали з регбі. Сенс гри полягає в тому, щоб взяти м'яч і підвести його до лінії поля суперника, долаючи опір. Але в регбі не можна кидати м'яч вперед. Коли ви побіжите і зрозумієте, що вас зараз зупинять, ви зможете повернути м'яч назад члену своєї команди. Він ловить його і біжить далі. Якщо він не може подолати опір, він біжить назад.

Сучасний Scrum створений двома людьми – Кеном Швабером та Джефом Сазерлендом. На офіційному сайті www.scrumguides.org ви можете ознайомитись з офіційним описом Scrum. На рисунку 2.23 зображено як виглядає загальна схема Scrum.

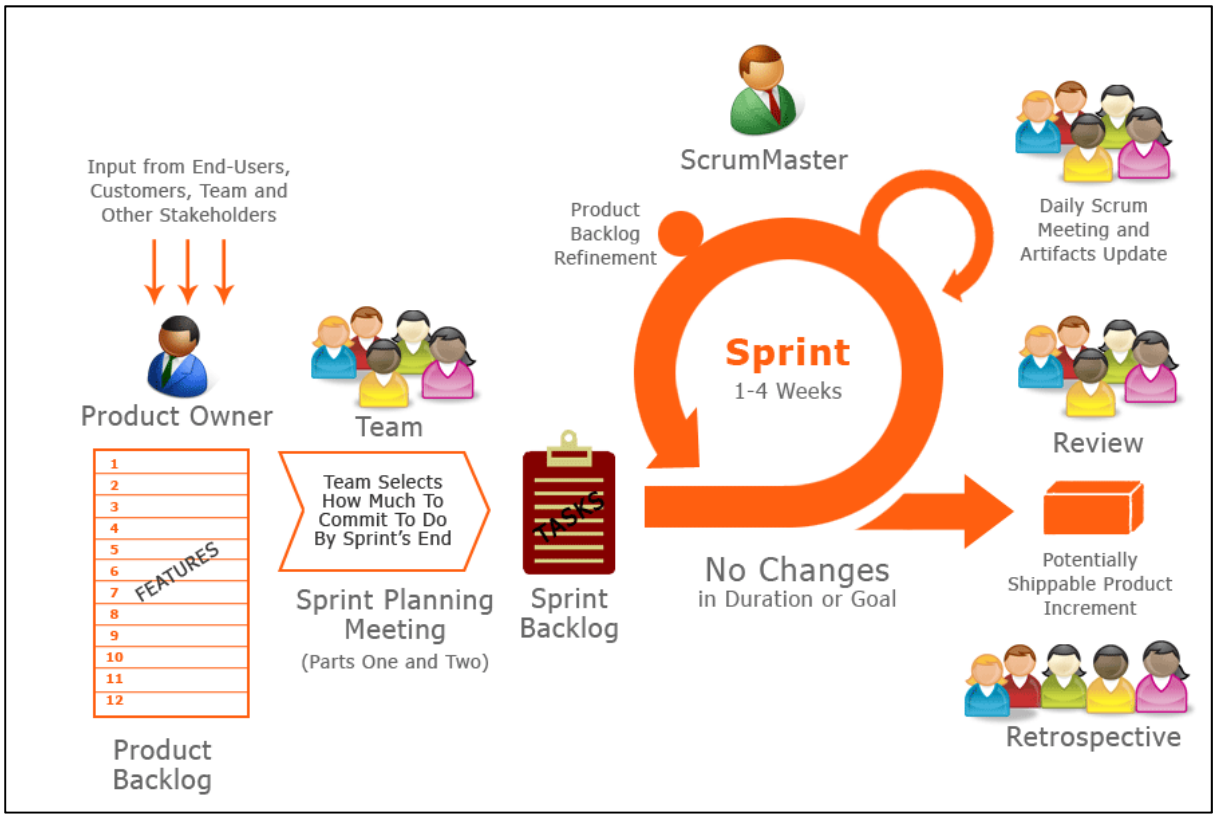


Рисунок 2.23 – Загальна схема Scrum

Отже, хочемо робити якийсь продукт. Він розрізаний на окремі шматочки, які називаються бэклогом (backlog) продукту. Це вимога. Тобто ми не маємо технічного завдання або якогось великого документа, який все заздалегідь описує. Зазвичай чим важливіші якісь вимоги, тим якісніше вони розбиті та описані. Цим займається власник продукту (product owner).

Середньостатистична команда складається з семи осіб (плюс-мінус дві особи). Якщо набагато менше, то, швидше за все, в команді не вистачає якихось фахівців, тому що мається на увазі, що Scrum-команда може самостійно зробити з беклог готовий продукт з новою функціональністю. Наприклад, у команді може не бути фронтенд-програміста. Якщо ви використовуєте Scrum і проєкт має на увазі фронтенд-розробку, то цього фахівця потрібно ввести до команди.

Scrum-команда складається з команди розробки, власника продукту та Scrum-майстра (рис. 2.24).

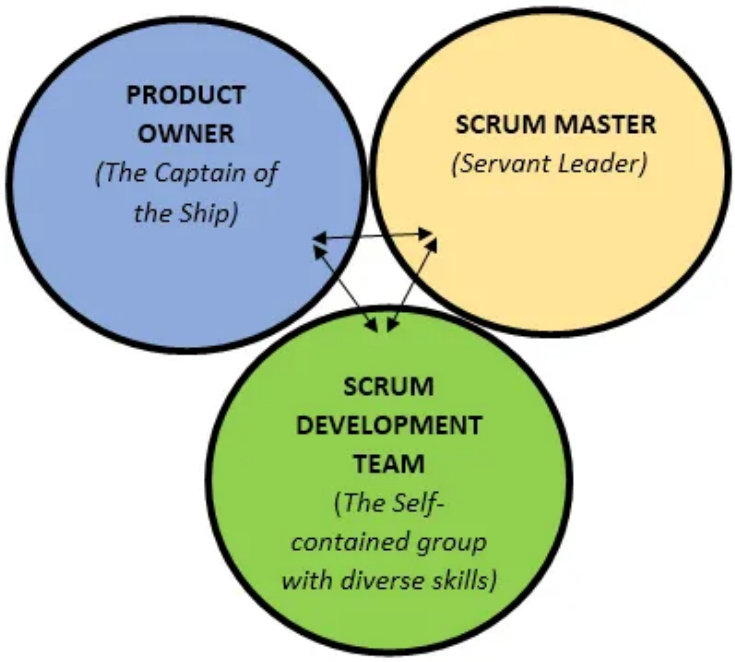


Рисунок 2.24 – Ролі в Scrum

Чому ми говоримо, що Scrum – це гнучкий Agile-фреймворк? Тому що він безпосередньо реалізує цінності та принципи, описані на початку публікації. Команда в Scrum має бути самоорганізованою, а Scrum-майстер допомагає їй стати такою, вчить як максимально швидко та ефективно створювати інкремент продукту, тобто нову версію програмного забезпечення з елементів резерву. Скрам-майстер відповідає за те, щоб усі процеси працювали, щоб учасники розуміли, чому і як це робиться.

Менеджер з продукту (product manager) – це особа, відповідальна за максимізацію цінності продукту, відповідальна за продукт.

Команда розробників має складатися з цільових професіоналів, які можуть під час кожного спринту робити реліз, тобто створювати готовий продукт на основі вимог.

Основні поняття в Scrum зображені на рис. 2.25.

Рисунок 2.25 – Основні поняття в Scrum

Щоденний scrum – це план, де учасники команди розповідають про те, що зроблено, з якими проблемами зіткнулися, що планується зробити найближчим часом, щоб усі розуміли, хто чим займається.

Спринт – це ітерація з фіксованим терміном, тобто Scrum має мати ритм.

Розберемо планування спринту. Перед спринтом всі збираються і вирішують, що робити під час спринту і в якій формі.

Огляд або демонстрація спринту: всі збираються і демонструють, що нового в інкременті продукту, дивляться, фіксують коментарі та, можливо, змінюють свої найближчі плани.

Ретроспектива – всі збираються і думають, що можна покращити в наступному спринті. Наприклад, було багато помилок, користувачі незадоволені. Спробуємо використати модульне тестування в наступному спринті. Спробуємо, в наступній ретроспективі побачимо, чи допомогло, ще щось придумаємо.

Беклог спринту – верхня частина беклог. Кількість пунктів зазвичай визначається швидкістю команди в заході, який називається «плануванням спринту», і виконується перед початком етапу спринту. Спринт триває 1–4 тижні (зазвичай 2 тижні). Чим швидше і дешевше ви зможете зробити реліз, тим менша тривалість цього етапу.

Кожного дня команда проводить 15-хвилинний stand-up, scrum-meeting, де всі члени команди синхронізують свої дії. Найчастіше ці зустрічі називають просто «скрами».

Після спринту йде демонстрація – «Огляд», сенс якого в тому, що команда показує роботу власнику продукту чи комусь іншому. Далі йде ретроспектива, в якій команда обговорює, що вийшло, а що – ні, аналізуються робочі процеси, атмосфера в колективі, намагаються щось покращити. Після цього починається наступний спринт. В результаті робота Scrum-команди може бути схожою на ланцюжок з багатьох спринтів.

Артефактами можуть бути картки на дошці з коротким описом, що являє собою конкретний функціонал. У цьому змісті картками може бути обговорення з власником продукту. Зазвичай це виливається в тікери в трекері, який ви використовуєте, – JIRA, Redmine тощо (найпопулярніші веб-сервіси для ведення проєктів на момент написання матеріалу).

Бэклог продукту – це всі квитки, картки чи інші вимоги як елементи беклогу, які є у вашому продукті.

Беклог спринту – найцінніші й пріоритетні вимоги.

Одним із прикладів Scrum-практик є оцінювання. Тут не буде канонічного/кошерного/ванільного Scrum'а, описаного Швабером і Сазерлендом. Говорячи про реальні практики, що використовуються командами, зауважимо, що є важкі, нібито точні методи, коли потрібно все «правильно» порахувати та зробити. Але майже всі великі проєкти вибиваються з термінів. І це стосується не лише IT-проєктів. Наприклад, був випадок, коли аеропорт не могли запустити в експлуатацію через те, що не був готовий софт, який відповідає за автоматичний розподіл багажу. Якщо ви використовуєте гнучкі методології, можете спокійно запускати проєкти. Коли мова йде про великі проєкти, то передбачити дату, коли буде готовий проєкт – це приблизно те саме, що підійти до команди і попросити назвати випадкове число.

Agile і Scrum пропонують такі практики: давати відносні оцінки, тобто «вимірювати в папугах». Це означає, що оцінюється час, який знадобиться на вирішення завдання, а також скільки знадобиться папуг. Зазвичай їх називають story point. Краще не прив'язувати ці story point до будь-яких проміжків часу. Як зробити таку оцінку, чому її називають відносною? Зазвичай беруть завдання, маленьке, стандартне і зрозуміле для всіх, і оголошують його мірою, еталоном – потрібно: 1 папуга, 1 story point. Взято наступне завдання, порівняно з першим воно в 2 рази більше. Скільки воно займає? 2 story point. І таким чином ми розкладаємо всі завдання. В результаті ми не оцінюємо кожне завдання в часі, а порівнюємо їх між собою. Якщо ми десь помиляємося, це нормально. Головне, щоб помилялися однаково в усіх завданнях. Оцінювання проводиться командою та в рамках команди.

Що можна зробити далі? Наприклад, виконати двотижневий спринт і виконати головні завдання без жодних оцінок. Коли спринт закінчиться, тоді отримаємо інкремент продукту, який міститиме низку завдань. Порахуємо, скільки story point набрано, і в наступному спринті можна просто запланувати таку ж кількість story point. Така оцінка відносна і емпірична. Не робиться так, як люблять робити керівники, вважаючи, що виконавець працює 8 годин з однаковою ефективністю, насправді вимірюється скільки команда може набрати story point за спринт. Шкала оцінок зазвичай підбирається для розмежування завдань різних класів (рис. 2.26).

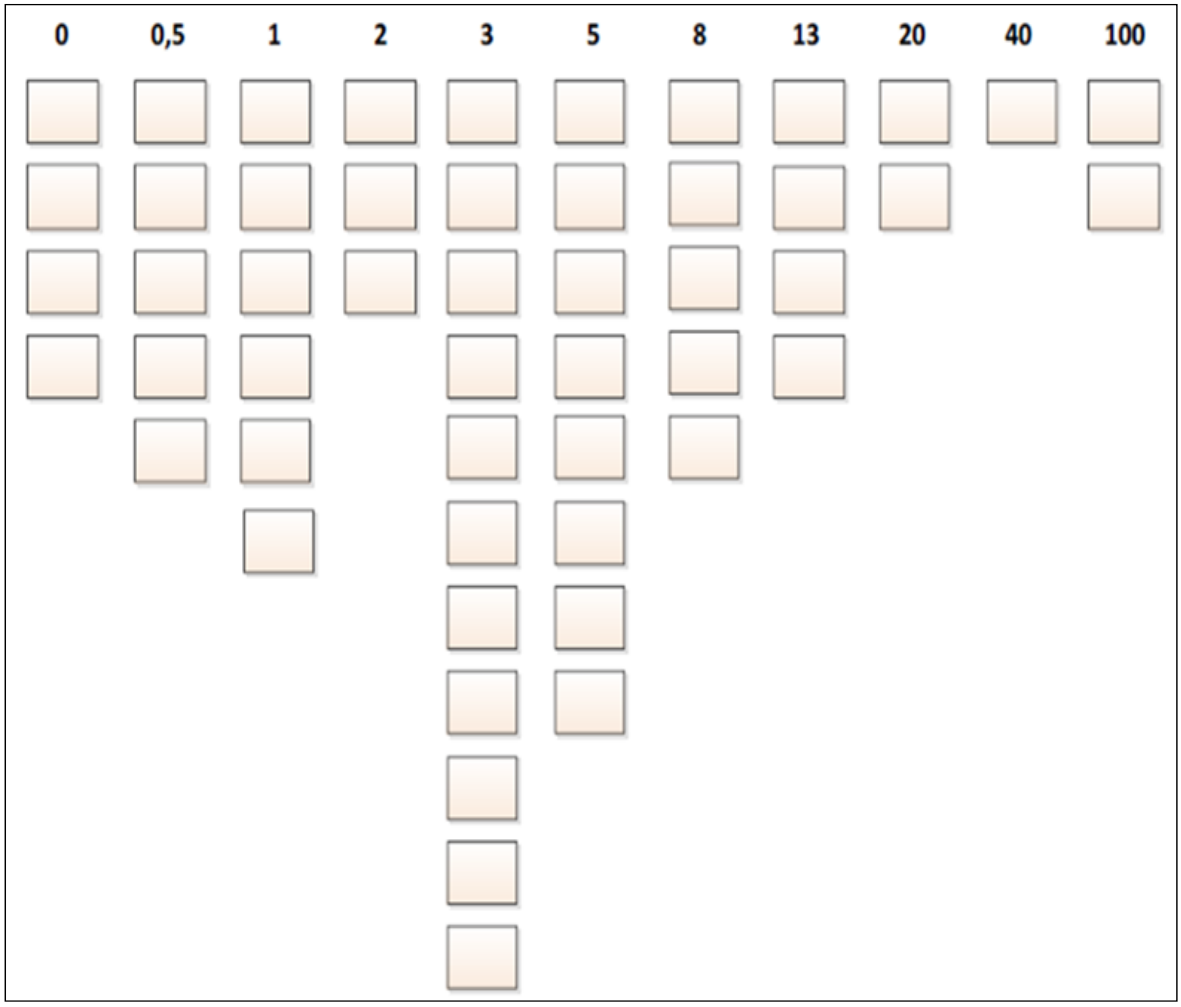


Рисунок 2.26 – Приклад карток для оцінювання в Scrum

Якщо придивитися уважно, це виглядає як числа Фібоначчі. При цьому зазвичай оцінюються дійсно великі завдання, які далі розбиваються на менші. Для формування оцінок зазвичай використовують покер-планування. Це модифікація методу Delphi, коли кожному учаснику команди видається колода карт з числами від 0 до 100. Також є картка з написом «Я не розумію, що це за завдання» та картка з кавою («Давайте перепочиньмо, я втомився»).

Після цього беремо певний номер завдання, читаємо і обговорюємо, що це таке: «Користувач вводить логін і пароль для того, щоб увійти на сайт». Ми обговорюємо, як відбувається ця авторизація, де ми зберігаємо дані користувачів. Потім кожен член команди кладе картку з оцінкою, яку він вважає за потрібне. При цьому різні учасники команди жодним чином не впливають один на одного. Після цього картки відкриваються та обговорюються (рис. 2.27).

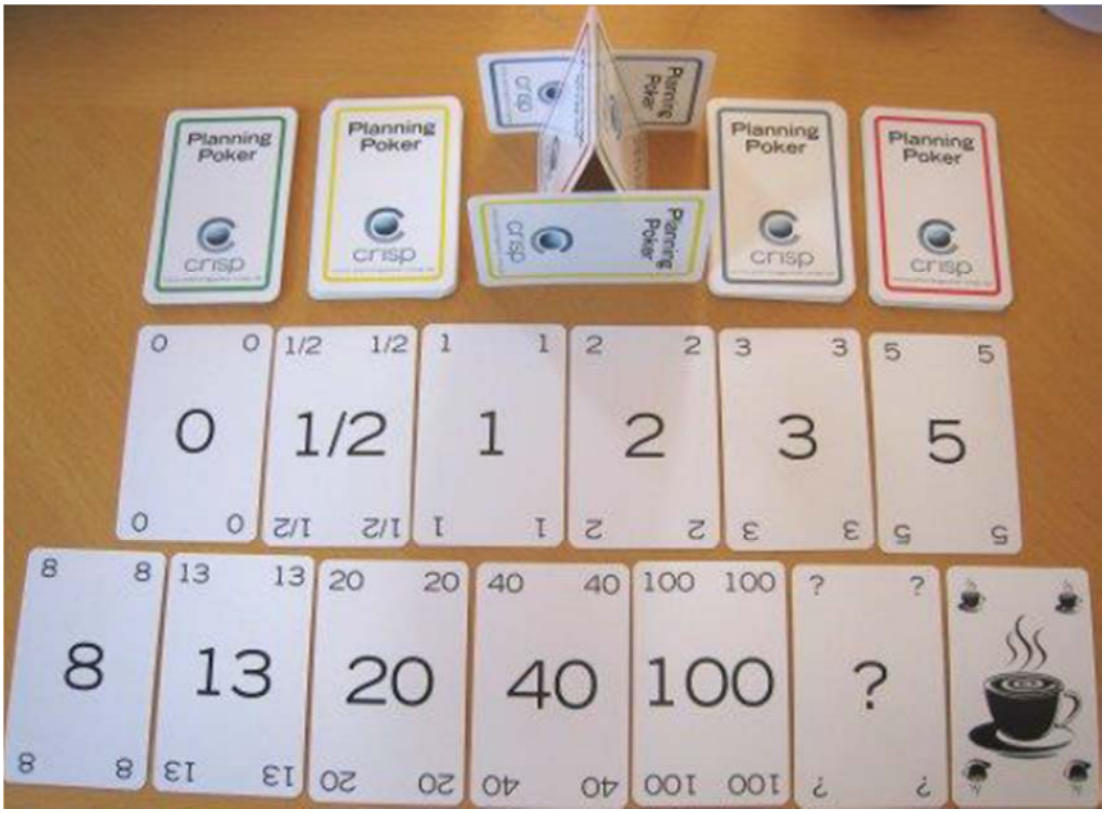


Рисунок 2.27 – Приклад карток для оцінювання в Scrum

За методом Delphi дискусія відбувається між тими, хто поставив найвищий і найнижчий бал. Той, хто поставив найбільше, зазвичай бачить деякі ризики, які інші члени команди не помічали. Він скаже: «Знаєте, ми давно не заходили в цю базу, де є логіни та паролі. Там потрібно щось переробити, додати стовпець, застосувати ще один хеш» і так далі. А той, хто поставив найнижчу оцінку, або не розуміє завдання, або бачить спосіб зробити це швидше, або вже зробив щось подібне. Далі йде другий етап обговорення: знову всі картки складаються, потім виставляються нові оцінки. Зазвичай оцінки більш-менш згладжуються. Знову йде етап обговорення, досягається консенсус. В результаті записується в трекер або прямо на картку, що у нас це завдання на 3 story point.

Чому дуже добрий Agile-підхід? Розмовляючи, обговорюючи зміст завдання, дії команди ґрунтувались на взаємодії людей, а не на якихось формальних процесах. Ще даний метод добрий тим, що тут виходить командна відповідальність за розмір завдання. Усі беруть на себе відповідальність за те, що завдання справді такого «розміру». Якщо ж ви вимірюватимете трудомісткість завдань, скажімо, у днях, то будуть ситуації, коли хтось у команді оцінить її у 8 днів і вона потрапить до нього, то він її і робитиме 8 днів.

Що робити, якщо замовник хоче зрозуміти, скільки часу займе реалізація проєкту і просить відразу дати йому оцінку? Ідеальний варіант – працювати з замовником за системою «час – матеріали». Якщо замовник новий, можна один проєкт зробити за Fixed Price, переконатися, що замовник

адекватний, і далі дотримуватися системи «час – матеріали». Якщо замовник не погоджується відразу на цю систему, то можна її скоригувати: наприклад, якщо ви закінчуєте проєкт до певної дати, то отримуєте якийсь бонус. На цю тему у мережі теж є презентації.

Розглянемо ще один приклад практики Scrum – швидкість команди. Беремо кожен спринт, вимірюємо, скільки story point зроблено. І якщо нам потрібно спланувати дев'ятий спринт, ми просто беремо середнє значення з попередніх спринтів. Це називається «принципом вчорашньої погоди» (рис. 2.28). Тут важливо те, що ми вибираємо найважливіші чи найцінніші завдання, виходячи зі швидкості роботи команди.

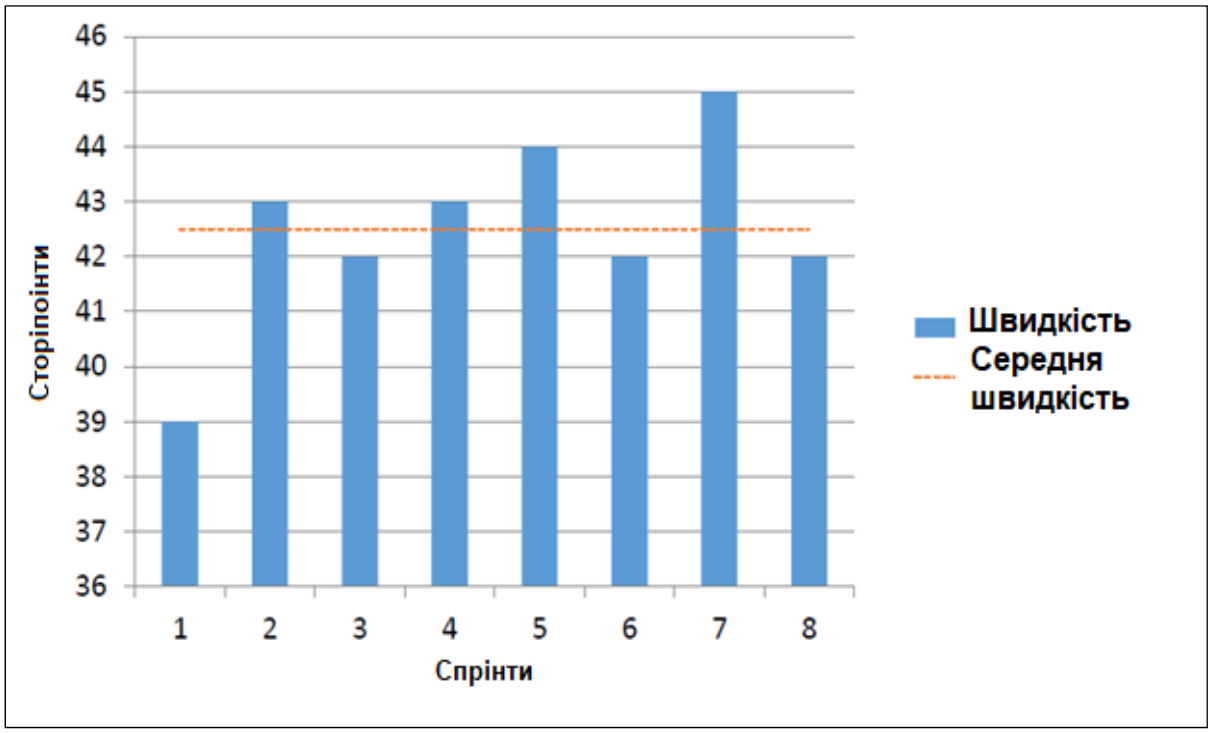


Рисунок 2.28 – Візуальне подання «принципу вчорашньої погоди»

Потрібно розуміти, що швидкість не буде рівномірною. Люди працюють зі змінною інтенсивністю, хтось хворіє, хтось повільніше через проблеми вдома, комусь потрібно терміново поспілкуватися в соцмережах, хтось пішов у відпустку. Завжди потрібно прямо й однозначно говорити замовнику продукту: «У нас є завдання A, B, C, ми встигнемо їх виконати точно, вони потрапляють у нашу найнижчу швидкість. Також є завдання D, яке ми, швидше за все, встигнемо виконати. Але є ще й завдання F, яке може випасти. Здається, що це неточність, але насправді, коли ви говорите замовнику, які саме найважливіші завдання будуть виконані, це підвищує довіру між вами та замовником продукту або клієнтом, і він зможе вибрати найважливіші завдання для кожного спринту та гарантовано їх отримає.

Ще одним прикладом Scrum-практик може бути колійний контроль. Як під час спринту контролювати, що у команди розробників усе йде гаразд? Команда спланувала спринт та розпочала його реалізацію. Для контролю використовується діаграма згорання Burn Down Charts (рис. 2.29).

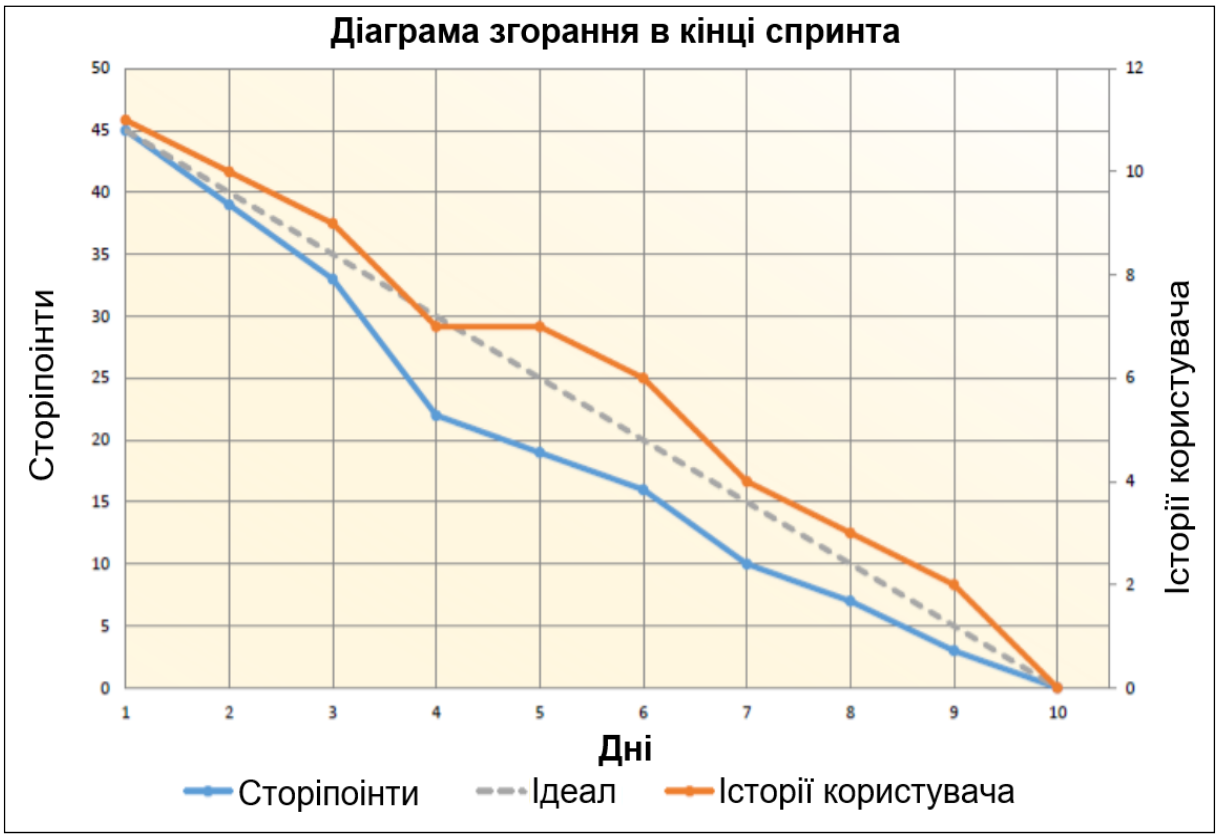


Рисунок 2.29 – Діаграма згорання Burn Down Charts

По горизонталі позначаємо дні – двотижневий спринт, 10 робочих днів. По вертикалі, з одного боку, відкладаються story point, з іншого боку, історії користувачів або елементи беклогу. Якби виконавці були роботами, а завдання були дрібними й розбитими на шматки, графік йшов би по пунктирній лінії – це лінія ідеального Burn Down Charts. Лінія зверху – скільки історій користувачів було зробилено, знизу – кількість story point. Такі графіки автоматично будуються в багатьох системах відстеження завдань.

Якщо графік знаходиться над лінією ідеального Burn Down, виконавці відстають, повільно працюють. Тоді до кінця спринту залишиться не нуль завдань або story point, а десь 20–25. Можна в Excel побудувати тренд або регресію і подивитися, скільки вийде задач з такою швидкістю дуже просто і наочно. Якщо команда бачить, що вона йде поверх ідеального Burn Down, то потрібно вживати заходів. Насправді зазвичай виходить так, як зображено на рисунку 2.30.

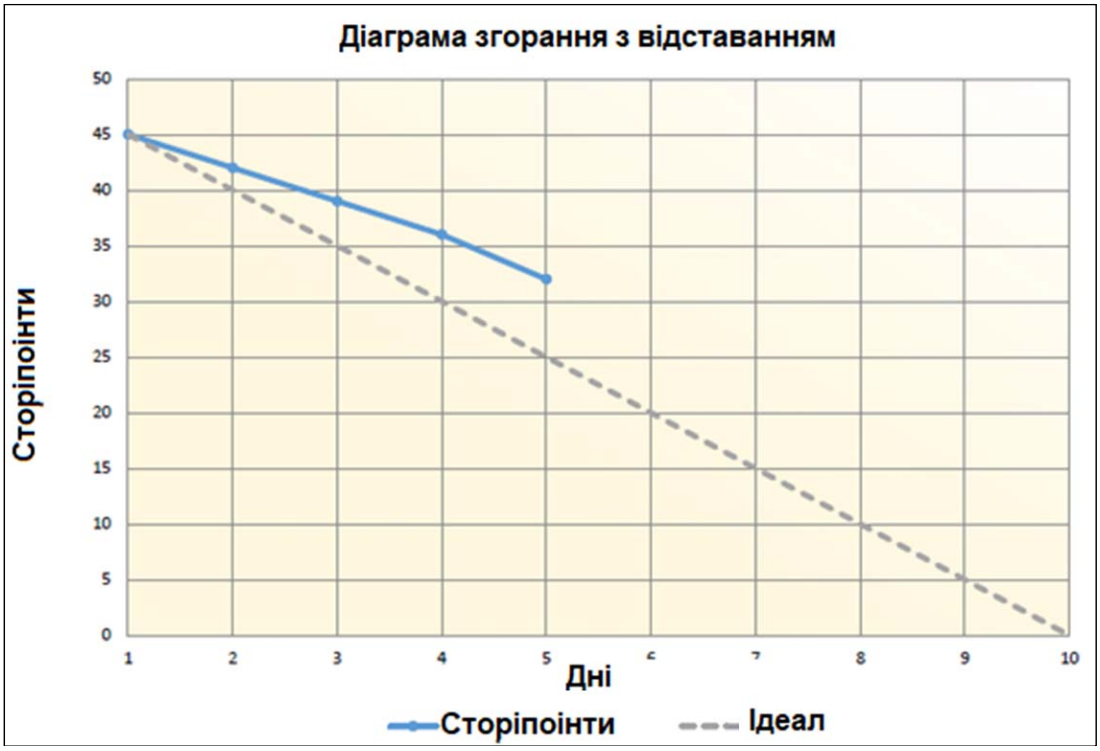


Рисунок 2.30 – Діаграма згорання з відставанням

Йдуть невеликі відставання, але це у хорошої команди. Інший варіант зустрічається рідше, коли Burn Down нижче за діагональну лінію. Це означає, що команда йде з випередженням, тобто, швидше за все, команда просто взяла мало завдань (рис. 2.31).

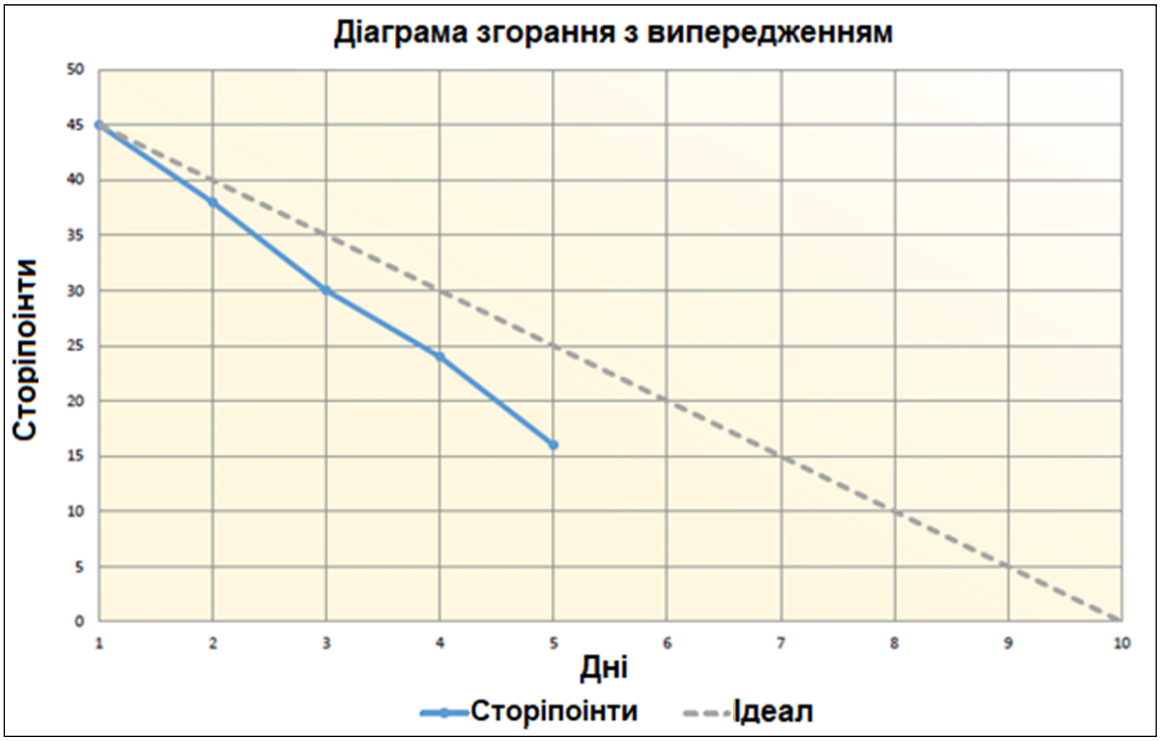


Рисунок 2.31 – Діаграма згорання з випередженням

Очевидно, що команді треба збільшити обсяг робіт. У крайньому випадку, тут ще можна зняти завдання, але це привід обговорити у ретроспективі, чому так вийшло. Цей графік – артефакт, який можна наприкінці двотижневої ітерації проаналізувати та зробити висновки.

Наступним прикладом Scrum-практик є дошки завдань. Зазвичай у Scrum використовують дошки завдань, хоча вони не є обов'язковим елементом. Команда розподіляє завдання по окремих етапах та розміщує на дошці у вигляді окремих карток. Існують електронні реалізації дошок завдань, плагіни для JIRA тощо. Завдання упорядковуються за рівнем важливості. Коли команда збирається зранку, оновлюються статуси завдань, їх переносять на інші етапи, дивляться, чи є десь затримки.

Іншим прикладом Scrum-практик є огляд спринту. По змозі, усі зацікавлені сторони беруть участь у цій зустрічі: розробники, користувачі, служба підтримки, системні адміністратори тощо. Щоб запустити цикл зворотного зв'язку, потрібен спринтовий огляд. Ви показуєте власнику продукту розроблений інкремент, він дивиться, робить зауваження, дає пропозиції, повертає їх вам. Ви берете в роботу, створюєте новий інкремент, через один–чотири тижні показуєте результат і знову отримуєте додаткові зміни до вимог. Тобто, ви запускаєте цикл, який, в кінцевому підсумку, приведе до продукту, який хоче отримати власник.

Також прикладом Scrum-практик є ретроспектива. Ретроспектива потрібна для постійних покращень. Гуру менеджменту Едвард Демінг колись сказав, що вдосконалюватись необов'язково, виживання – справа добровільна. Ретроспектива – саме той етап, на якому ви можете зайнятися вдосконаленням. Як це відбувається? Вся команда збирається та обговорює всі щаблі до самої ретроспективи. Зазвичай це триває від години до чотирьох, може тривати навіть день. Якщо ви подивитеся олдскульні книжки, там є ретроспектива навіть на кілька днів.

Починається ретроспектива з відкриття. Зазвичай, вона займає 5% часу. Завдання – розрухати всіх присутніх на ретроспективі, тому що дуже часто в командах, особливо айтішних, присутні не дуже товариські люди, але мають блискучі думки. Завдання ведучого у тому, щоб розговорити цих людей. Другий етап – збирання даних, він займає до половини часу. Команда шукає якісь факти. Їх можна згадати, дістати з трекера, роздрукувати. Також можна зібрати статистику з багів, хто репортував, який їх статус – безліч варіантів. Після збирання фактів починається мозковий штурм: потрібно зрозуміти, у чому проблема, проникнути в суть, згенерувати ідеї, які допоможуть її вирішити. На це йде до третини часу. Наприклад, якщо у команди багато дрібних багів, що виникають не на рівні інтеграції системи, а в коді розробників, можна запропонувати використовувати модульне тестування. Якщо дуже погана якість, як її покращити? Можна спробувати парне програмування, якісь інструменти, які роблять автоматизовану перевірку коду, ще щось. Зазвичай набирається 5–10 ідей. Далі треба втілити ці ідеї у життя. Не можна одразу впровадити code review, ро-

зробити якісь інструменти. Тому вибирається максимум одна–дві ідеї, реалізацію яких потрібно запланувати на наступний спринт. Після цього закривається ретроспектива. І ще через два тижні можна оцінити, вдалося досягти потрібного результату чи ні, вивчити нові проблеми.

На ретроспективі також можна зрозуміти моральний дух команди – для цього теж є інструменти. Можна просто накреслити тимчасову лінію спринту, щоб кожен член команди згадав, що відбувалося цього дня, наклеїв стікер із фактом «закінчив розробляти завдання», «виправляв поганий код», «застосував нову технологію Machine learning». Після цього за кожним фактом можна знизу нарисувати, наскільки цей факт був для людини цікавим, або навпаки. Після цього побудувати медіану, яка відобразить стан та динаміку морального духу команди.

Є таке поняття, як «Цикл Демінга» (рис. 2.32). Він складається з чотирьох етапів:

– Планування (Plan);
– Реалізація (Do);
– Перевірка (Check);
– Зміна (Act).

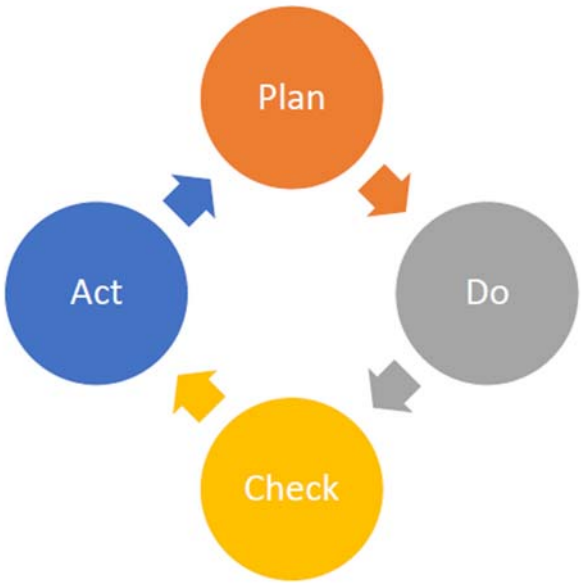


Рисунок 2.32 – Цикл Демінга

Під час ретроспективи можна створити план, який буде надалі змінюватись. Наприклад, впроваджується модульне тестування (unit-тестування) – як впроваджується, який інструмент використовується, яке тестування потрібно отримати. Потім настає фаза впровадження (як правило, це спринт, якщо ми говоримо про Scrum), коли ми реалізуємо рішення. У наступній ретроспективі ми зможемо перевірити, чи вдалося нам досягти бажаного ступеня охоплення тестування. Ми можемо побачити, чи зменшилася кількість помилок у тих місцях, які ми охопили тестами.

Після цього можна вносити зміни: наприклад, ми хотіли зробити охоплення 50% – зробили, кількість багів зменшилась, але вони все одно залишилися – збільшимо до 70%. Або зробивши 70%, прокрутивши цикл вдруге, перевірили – покращилось. Зробимо 90%. Знову прокрутили, кількість помилок не зменшилася, а вартість написання та підтримки тестів висока. Отже, потрібно дещо зменшити. Завдяки цьому циклу команда поступово вдосконалює деякі процеси. Найпростіший варіант ретроспективи – Real-Time Board Service (рис. 2.33).

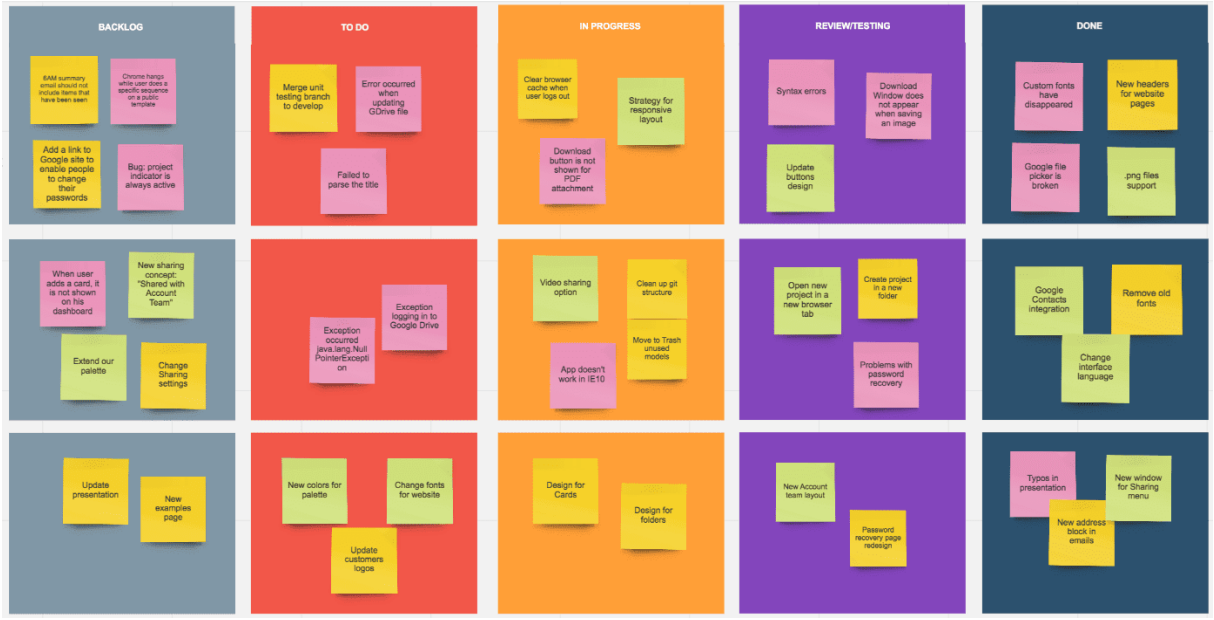


Рисунок 2.33 – Приклад Real-Time Board Service

Команда вішає стікери, що потрібно зробити, що зроблено, що потрібно поліпшити тощо. Тут можуть бути як думки вголос: «Все зробили. Це дуже круто», «Самостійна команда», «Команда маленька – не подобається», так і технічні речі. На основі цього можна висунути ідеї та деякі з них відібрати на реалізацію.

Потрібно сказати, що Scrum, як і всі гнучкі методології, найкраще працює у командах, які сидять в одній кімнаті. Проте в мережі можна знайти сотні презентацій про те, як застосовувати гнучкі методології у розподілених командах, коли люди працюють на відстані. Тут ідея така: замість реального спілкування максимально використовувати програмні та апаратні інструменти. Що зазвичай використовують? По-перше, спільний трекер, щось на зразок JIRA. Це справді допомагає. Популярні програмістські чати, наприклад, HipChat. Для спілкування – Skype, Hangout. Головне, щоб був відеозв'язок і щоб можна було демонструвати екрани своїх комп'ютерів.

Другою за популярністю методикою є Kanban. Окремі компанії працюють одночасно за Scrum і Kanban, виходить Scrumban. Напевно, це один із майбутніх трендів. Історична довідка: Kanban з'явився у Японії. Цим

словом називався папірець із пул-запитом на якісь дії. Наприклад, потрібна якась деталь, на неї робиться окремий канбан. Але в IT це все-таки застосовується трохи в іншому вигляді.

Цінності та принципи. В айтішному вигляді Kanban з'явився в 2010 році, тобто це досить свіжа, добре описана методологія. Її автор – Девід Андерсон. Якщо Scrum передбачає жорстко прописані процеси, які мають зламати те, що було в організації до цього, тобто «Так, всі ми тепер працюємо за спринтами, з ранку приходимо, стендапимося, наприкінці спринту показуємо демонстрацію», то Kanban має на увазі більш еволюційні зміни.

– Починаємо з того, що є зараз, і далі шляхом еволюції та поступових змін робимо з цього незрозумілого хаосу чітко налаштовану Kanban-систему. При цьому намагаємося лише еволюційно змінювати ролі, зони відповідальності та обов'язки. Заохочуємо ініціативні дії на всіх рівнях організації. Головна практика – візуалізація, зазвичай у вигляді дошки. Робота кожного члена команди має бути візуалізована, її мають бачити всі.

– Кількість роботи у кожному процесі обмежується. Тобто, в роботі одночасно може бути не більше якоїсь кількості завдань. Це потрібно для унеможливлення мультитаскінгу, який вбиває ефективність. До того ж, це дає певні інструменти управління потоком завдань.

– Усі правила мають бути явними. Потрібно дати означення завершеності. Наприклад, завдання виконано, якщо написаний код, є unit-тести з покриттям 70% тощо.

– Необхідно робити покращення за допомогою постійних експериментів, використовуючи моделі та науковий підхід, зокрема, цикл Демінгу.

Приклад візуалізації показано на рисунку 2.34.

Рисунок 2.34 – Приклад візуалізації в Kanban

Зазвичай використовується та сама дошка, що і в Scrum. Найпростіший варіант – прото-Kanban. Потік завдань розбивається на окремі етапи.

Щось у плані, щось в аналітиці, щось у розробці, щось у тестуванні, щось вже зроблено. При цьому реалізується принцип обмеження кількості завдань, що одночасно перебувають у роботі – WIP (Work in Progress). Є формула Літтла, яка пов'язує швидкість проходження завдання у такій системі та кількість одночасних завдань. Що менше WIP, то швидше завдання проходять ланцюжок. Припустимо, проєкт має завал у тестуванні, а розробник зробив наступне завдання. Він бачить, що у тестувальників проблема і тоді розробник допомагає їм щось зробити, або вони йдуть до керівника та кажуть: «Нам потрібен ще тестувальник».

Зазвичай команда починає з великого обмеження завдань у роботі, наприклад, не більше двох завдань на людину. Якщо команда має одного тестувальника – два завдання для розробки, якщо чотири тестувальника – вісім завдань. Поступово загальна кількість завдань зменшується, швидкість роботи зростає. І дошка вже виглядає приблизно так, як зображено на рисунку 2.35.

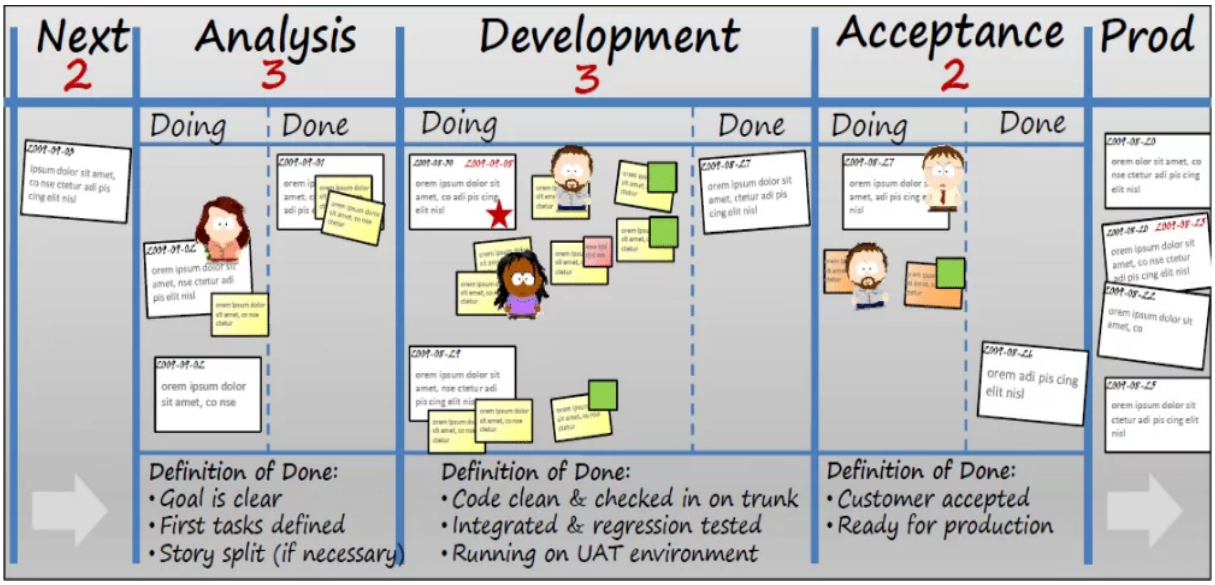


Рисунок 2.35 – Приклад дошки Kanban

Тут є ті ж WIP, внизу – критерії готовності (Definition of Done). Стовпець ділять на дві частини: «в роботі» (Doing) та «виконане» (Done). Іноді дошку ділять на доріжки та розміщують WIP по горизонталі. Це вже просунута, повноцінна Kanban-система. Кожна доріжка відповідає певному класу обслуговування. Наприклад, є гарячі завдання, які треба зробити швидше, тоді це окремий клас обслуговування, під нього варто забронювати WIP.

Як і в Scrum, тут також можна створювати діаграми. Зазвичай їх називають «Діаграми кумулятивного потоку» (рис. 2.36). По горизонталі відзначено час, по вертикалі – кількість завдань. Різними кольорами показано різні етапи. Покращення потрібно здійснювати на основі цифр, використовуючи науковий підхід. Ці цифри можна отримати з діаграм. Найважливішими з них є WIP, тобто кількість завдань за винятком запланованих і виконаних. Ми його маємо скорочувати.

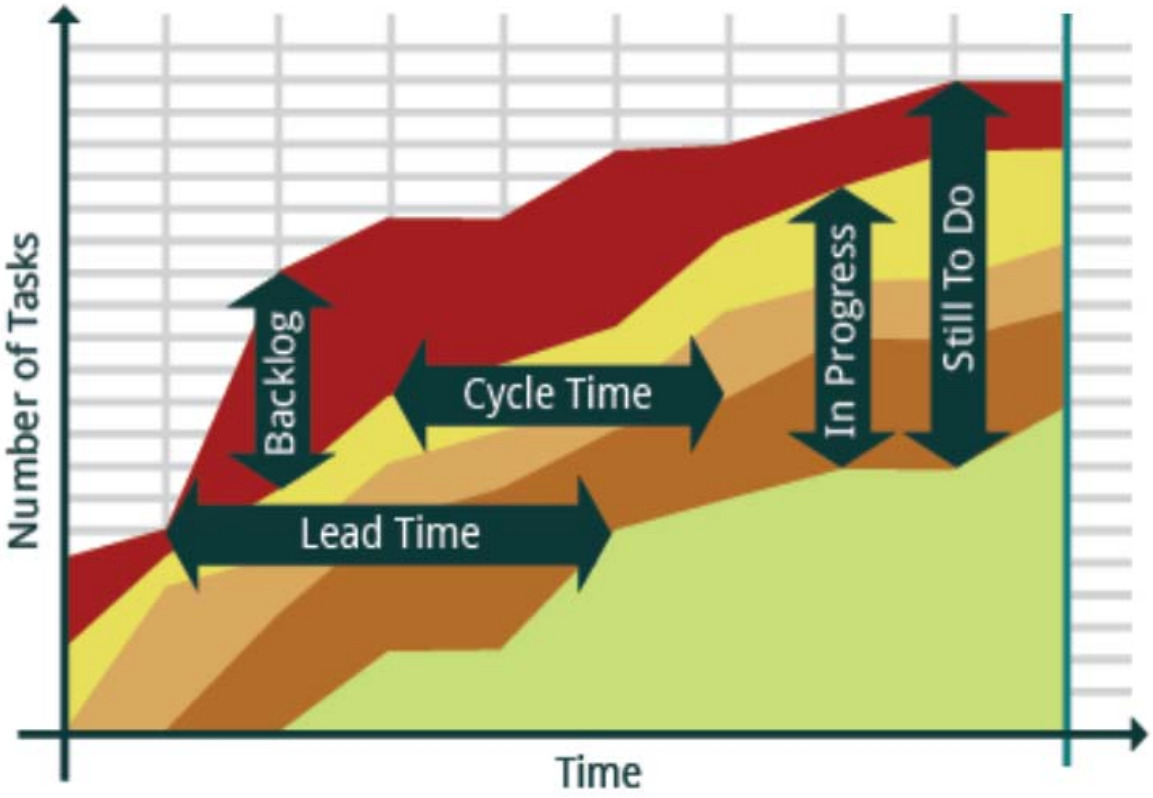


Рисунок 2.36 – Приклад діаграми кумулятивного потоку

Інші важливі критерії – Cycle Time та Lead Time. Визначення бувають різними, треба дуже пильно дивитися. Ці два числа показують, як швидко завдання проходять через вашу Kanban-систему.

В даному випадку Lead Time охоплює очікування, тобто, як сприймають вашу Kanban-систему замовники. Cycle Time – наскільки швидко завдання проходить через Kanban-систему без очікування, у загальному беклогу. Обидва параметри потрібно зменшувати, тоді ваша система працюватиме швидше.

Kanban дуже добре приживається у компаніях із корпоративною командною культурою, коли є якась ієрархія. Scrum зручний для команд, які вже добре спілкуються, у компаніях із плоскою структурою, де мало начальників.

Контрольні питання

1) Який існує взаємозв'язок між процесами управління?

2) Які групи процесів Ви знаєте?

3) За якими критеріями можна визначити основні чи допоміжні процеси?

4) Назвіть особливості основних моделей управління.

5) В чому полягає суть ведення документації проєкту?

6) Як оцінити ризики проєкту?

7) Які основні інструменти управління ризиками проєкту?

8) Які відомі програмні засоби для управління проєктами Ви знаєте?

9) Навіщо потрібна методологія гнучкої розробки?

10) Навіщо застосовується методологія Agile?

11) Які з 12 принципів Agile варто виділити? Чому саме ці?

12) Які основні інструменти Agile або Scrum є найбільш дієві?

13) Від яких факторів залежить швидкість роботи команди?

3 МОДЕЛЮВАННЯ БІЗНЕС-ПРОЦЕСІВ

3.1 Функціональне моделювання SADT (IDEF0)

Більшість з тих, хто займаються реалізацією проєктів, пов'язаних зі створенням або розвитком корпоративних інформаційних систем, згодні з тезою, що замовнику потрібна інформаційна система, яка підвищує ефективність діяльності підприємства. Однак замовники і розробники інформаційних систем до сих пір розмовляють різними мовами: вони по-різному розуміють «що значить підвищити ефективність підприємства».

Розробники інформаційних систем дуже часто під підвищенням ефективності розуміють зростання кількості робочих станцій у локальній обчислювальній мережі (ЛОМ) підприємства, зростання пропускної спроможності ЛВС, зростання кількості документів, обробка яких здійснюється на автоматизованих робочих місцях (АРМах) тощо.

Функціональна модель системи базується на функціональній схемі. На додаток до суто функціональних діаграм IDEF0 (Icam DEFinition), ця модель може містити діаграми, керовані даними, а саме DFD і IDEF3. Отже, функціональна схема може містити такі діаграми:

- функціональні діаграми IDEF0;
- діаграми потоків даних DFD (Data Flow Diagramming);
- діаграми опису послідовності процесів IDEF3 (Work Flow Diagramming);
- діаграма дерева вузлів функціональної моделі (Node Tree Diagramming).

У функціональній моделі головну роль відіграють діаграми IDEF0. Діаграми DFD (потоки даних) і IDEF3 (опис послідовності процесів) мають тенденцію доповнювати модель на нижніх рівнях декомпозиції, хоча вони можуть бути автономними та створюватися як окремі діаграми, починаючи з верхнього рівня. Діаграма дерева вузлів є демонстрацією, вона показує модель в цілому.

Для того, щоб замовник і розробник інформаційної системи розуміли один одного, потрібно, щоб розробник переорієнтувався з вирішення технічних завдань зі створення чи розвитку інформаційної системи на рішення комплексних завдань з підвищення ефективності діяльності підприємства замовника. При такому підході на перший план виступає проблема ефективного способу вивчення сфери діяльності замовника:

- обстеження існуючої бізнес-архітектури, ділових процесів, бізнес-правил, інформаційних потоків;
- ідентифікація проблем, «вузьких» місць, що негативно впливають на ефективність діяльності підприємства;
- розробка та реалізація заходів усунення наявних проблем і зміни бізнес-архітектури підприємства, перебудови ділових процесів;

– розробка конкретного проєкту корпоративної інформаційної системи, реалізація цього проєкту та супровід в майбутньому.

В ході реалізації програми інтегрованої комп'ютеризації виробництва (ICAM), запропонованої свого часу для аерокосмічної промисловості США, була виявлена потреба в розробці методів аналізу взаємодії процесів у виробничих системах. Для задоволення цієї потреби була розроблена методологія IDEF0 (Integrated Definition Function Modeling), яка в даний час прийнята як федеральний стандарт США. Методологія успішно застосовувалася в різних галузях, продемонструвавши себе як ефективний засіб аналізу, проєктування та подання ділових процесів. В даний час методологія IDEF0 широко застосовується не тільки в США, а і у всьому світі. В Україні IDEF0 успішно застосовувався в державних установах (наприклад, у Державній Податковій Інспекції), в комерційних банках, на підприємствах.

Методологія IDEF0 заснована на концепції блока, який відображає деяку бізнес-функцію. Чотири сторони блока виконують різні ролі: ліва означає «вхід», права – «вихід», верхня – «управління», нижня – «механізм» (рис. 3.1).

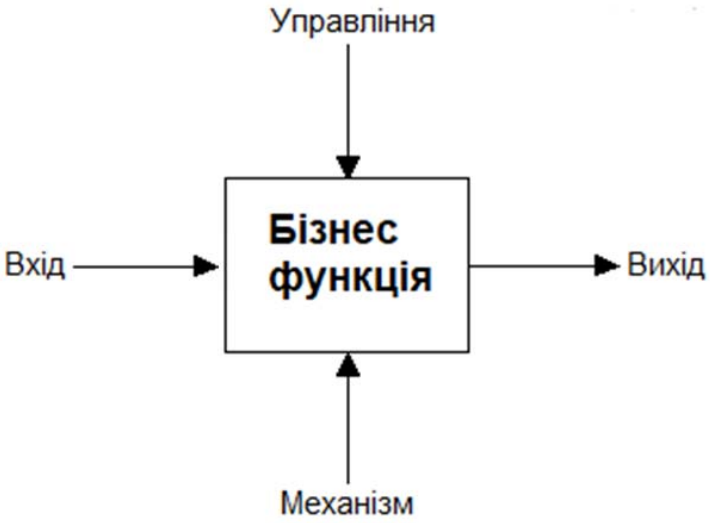


Рисунок 3.1 – Загальний принцип методології IDEF0

Розглянемо загальні характеристики моделі IDEF0. Ця модель, як ми вже знаємо, описує функції системи, тобто, як система досягає своїх цілей, які процеси в ній відбуваються, як ці процеси взаємопов'язані. Модель є деревоподібною топологічною структурою і створюється на основі функціональної декомпозиції цілей і завдань системи. Розберемось, що це означає. По-перше, функції системи описуються в загальних рисах без деталізації. Цей опис відповідає так званій контекстній діаграмі. Вона має багато спільного з моделлю системи типу «Чорний ящик», але вказує не назву системи, а її основну функцію (ціль, завдання). Надалі контекстна діаграма піддається функціональній декомпозиції. В результаті останнього, основна функція системи, описана за допомогою контекстної діаграми, поділяється

на підфункції (ціль – підціль, задача – підзадача). Кожну підфункцію потім розбивають на менші підфункції і так далі, поки не буде досягнуто найпростішого рівня деталізації. Результатом функціональної декомпозиції є поділ функцій системи на елементарні процеси з точки зору аналізу, які можна описати простими специфікаціями. Результат функціональної декомпозиції можна подати у вигляді ієрархічної моделі, яку називають деревом вузлів функціональної моделі (Node Tree Diagramming). Дерево вузлів є каркасом функціональної моделі, але не самою функціональною моделлю. Ця діаграма показує загальний склад моделі. Її зовнішній вигляд показано на рис. 3.2.

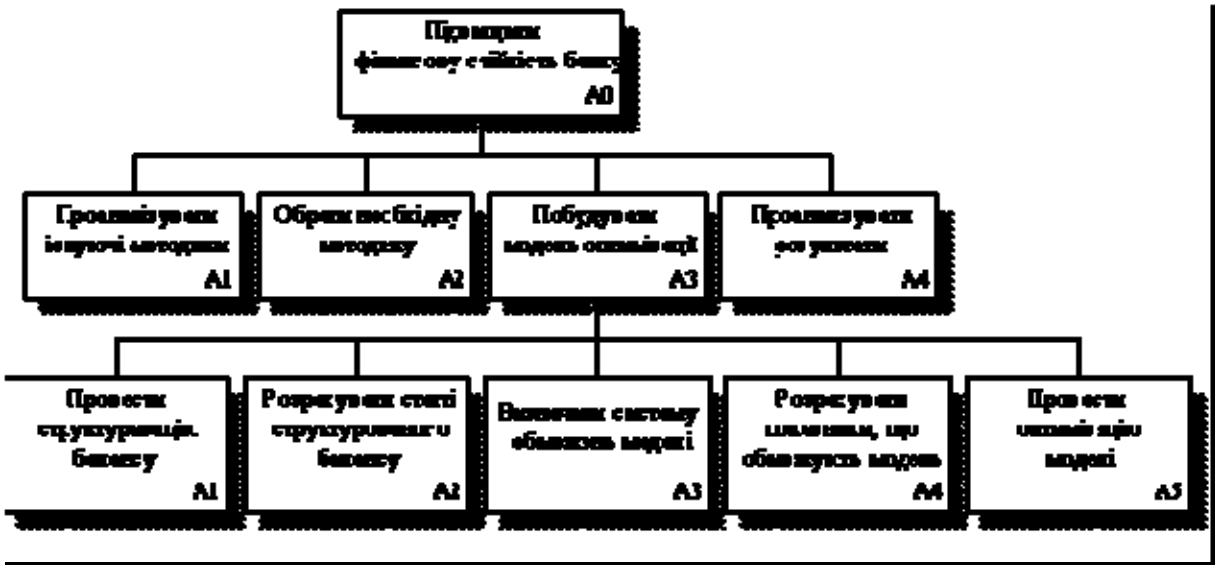


Рисунок 3.2 – Приклад дерева вузлів функціональної моделі

Контекстна діаграма зображається прямокутником з вхідними та вихідними значеннями. На відміну від моделі Black Box, у прямокутнику контекстної діаграми вказується не назва системи, а її основна функція (призначення системи). Входи та виходи контекстної діаграми розподіляються не на дві, а на чотири сторони прямокутника. Призначення їх такі:

– Ліва частина відповідає входам системи (input), величинам, які надходять в систему і обробляються нею у вихідні значення;

– Верхня сторона відповідає входам для контролю (control), тобто різноманітним керувальним діям, командам, стратегіям поведінки, процедурам, документам, що регламентують виконання роботи, тощо. Ці значення не змінюються, а служать лише для керування;

– Права частина відповідає виходам системи (output), продуктам її діяльності, результатам перетворення вхідних величин, шкідливим виділенням, відходам тощо;

– Нижня сторона відповідає механізмам, а саме: засобам і ресурсам, за допомогою яких виконуються функції, зазначені в прямокутнику.

Зовнішній вигляд контекстної діаграми показано на рис. 3.3.

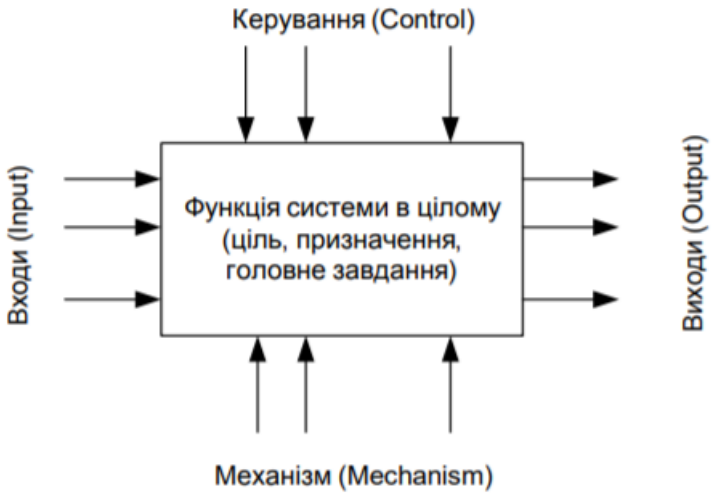


Рисунок 3.3 – Загальний вигляд контекстної діаграми

У моделі IDEF0 є п’ять типів з’єднань функціональних блоків. Кожному з них відповідає певне розташування дуг відносно блоків. Типи з’єднань бувають (рис. 3.4):

– вхід – вихід;
– керування;
– вихід – механізм;
– зворотний зв’язок по керуванню;
– зворотний зв’язок по входу.

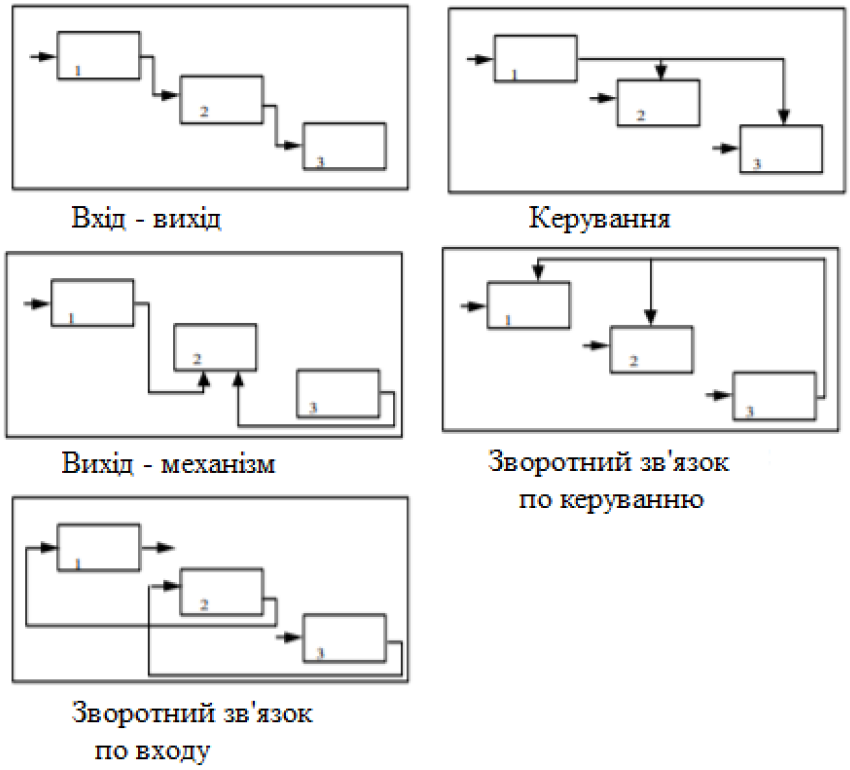


Рисунок 3.4 – Типи зв’язків блоків функціональної моделі IDEF0

Фрагмент реальної моделі зі зворотним зв’язком щодо керування показано на рис. 3.5.

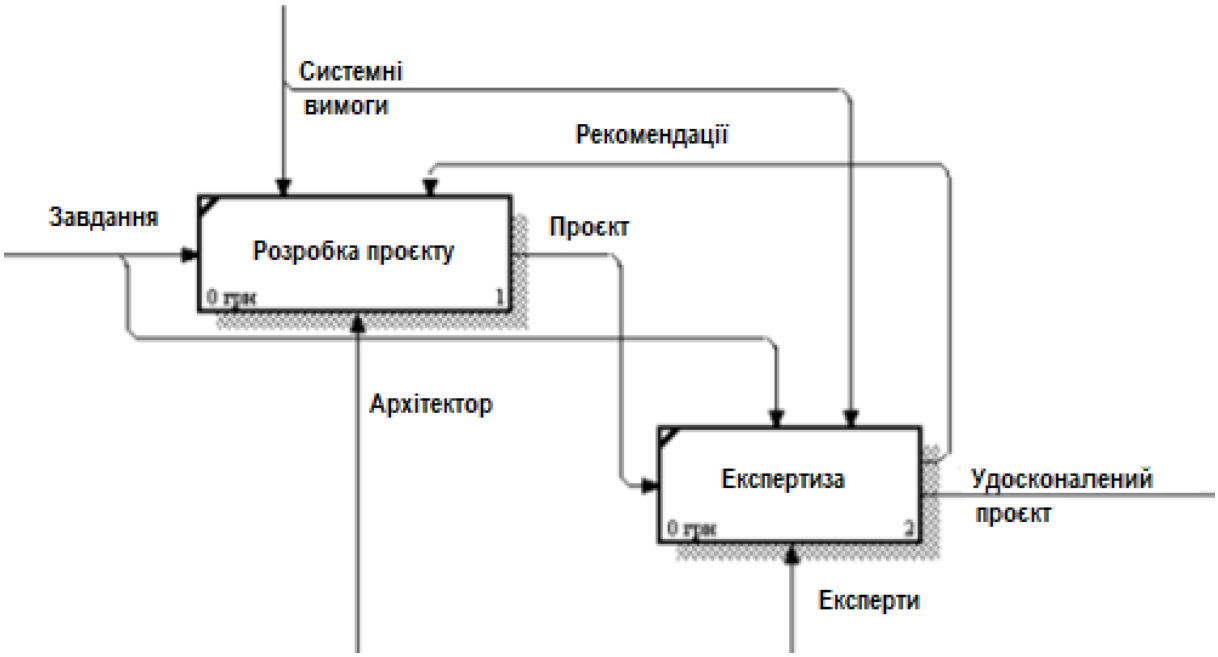


Рисунок 3.5 – Зворотний зв’язок щодо керування

Функціональна модель будується на кількох сторінках і друкується на аркушах паперу. На першому аркуші будується контекстна діаграма, на другому аркуші – діаграма декомпозиції контекстної діаграми, на наступних аркушах – діаграми декомпозиції кожного функціонального блока. Таким чином, вся модель складається з низки діаграм декомпозиції функціональних блоків. На додаток до діаграм декомпозиції, модель може також містити діаграму вузлів функціональної моделі, окремі діаграми, які називаються діаграмами експозиції (іменуються FEO-діаграмами), звіти для всієї моделі, стрілки, блоки, аналіз функціональних витрат та інші документи. Усі документи оформляються за певними правилами, нумеруються, об’єднуються в один документ. Комп’ютерна версія моделі зберігається в пам’яті комп’ютера і може бути збережена як окремий файл з розширенням * .bp1.

IDEF0 реалізує три основні принципи моделювання процесів:

– принцип функціональної декомпозиції;

– принцип обмеження складності;

– принцип контексту.

Принцип функціональної декомпозиції – це спосіб моделювання типової ситуації, коли будь-яку дію, операцію, функцію можна розкласти (декомпозувати) на простіші дії, операції, функції. Іншими словами, складну бізнес-функцію можна подати як набір елементарних функцій. Подаючи функції графічно, у вигляді блоків, можна зазирнути всередину блока та детально розглянути його структуру та склад (рис. 3.6).

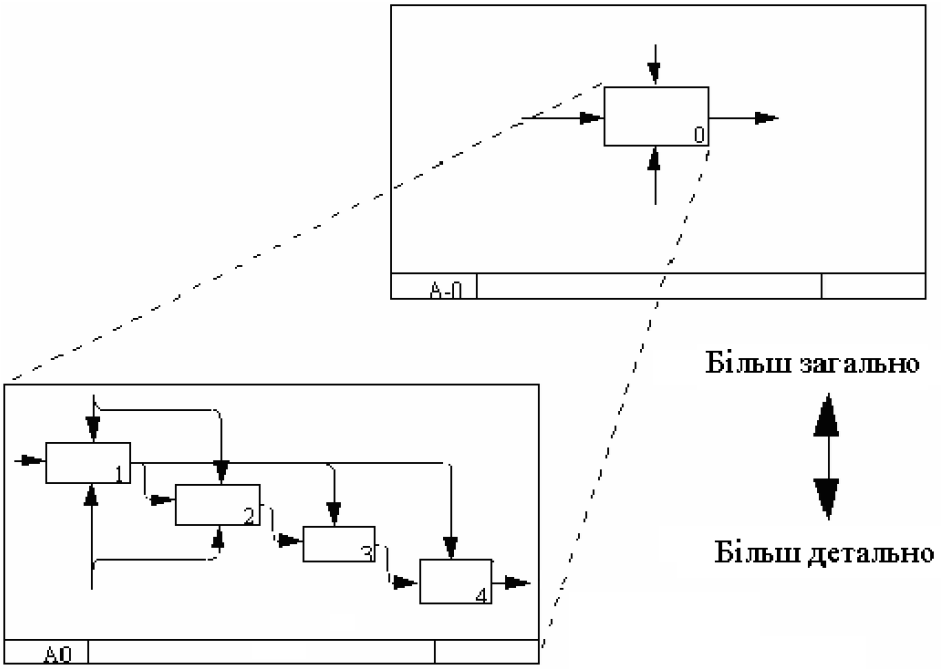


Рисунок 3.6 – Графічне подання структури в IDEF0

Після ознайомлення з основними поняттями та принципами функціонального моделювання бізнес-процесів виникає закономірне питання: як це сприяє підвищенню ефективності та якості роботи підприємства.

Побудова моделі «ЯК Є». Опитування підприємства є обов'язковою частиною будь-якого проєкту зі створення або розвитку корпоративної інформаційної системи. Побудова функціональної моделі «ЯК Є» дає можливість чітко зафіксувати, які бізнес-процеси здійснюються на підприємстві, які інформаційні об'єкти використовуються при виконанні бізнес-процесів та окремих операцій. Функціональна модель «ЯК Є» є відправною точкою для аналізу потреб підприємства, виявлення проблем і «вузьких місць» та розробки проєкту покращення бізнес-процесів.

Бізнес-правила. Модель бізнес-процесу дозволяє виявити і точно визначити бізнес-правила, які використовуються на підприємстві.

На рисунку 3.7 наведено фрагмент функціональної моделі. Функціональна модель дозволяє не тільки ідентифікувати існування правила, але також визначити, при виконанні якої операції і на якому робочому місці воно має застосовуватися.

Дуже часто правила ведення бізнесу на підприємстві не прописані в інструкції: вони ніби є, а ніби їх немає. Таким чином, спроби щось змінити в діяльності підприємства чи підрозділу можуть бути невдалими лише тому, що ці зміни суперечать усталеним правилам ведення бізнесу.

Інформаційні об'єкти. Функціональна модель дозволяє ідентифікувати всі інформаційні об'єкти, якими оперує підприємство у своїй діяльності. На відміну від інформаційних моделей (Data Flow Diagrams, IDEF1X), функціональна модель IDEF0 показує, як саме інформаційні об'єкти використовуються в бізнес-процесах.

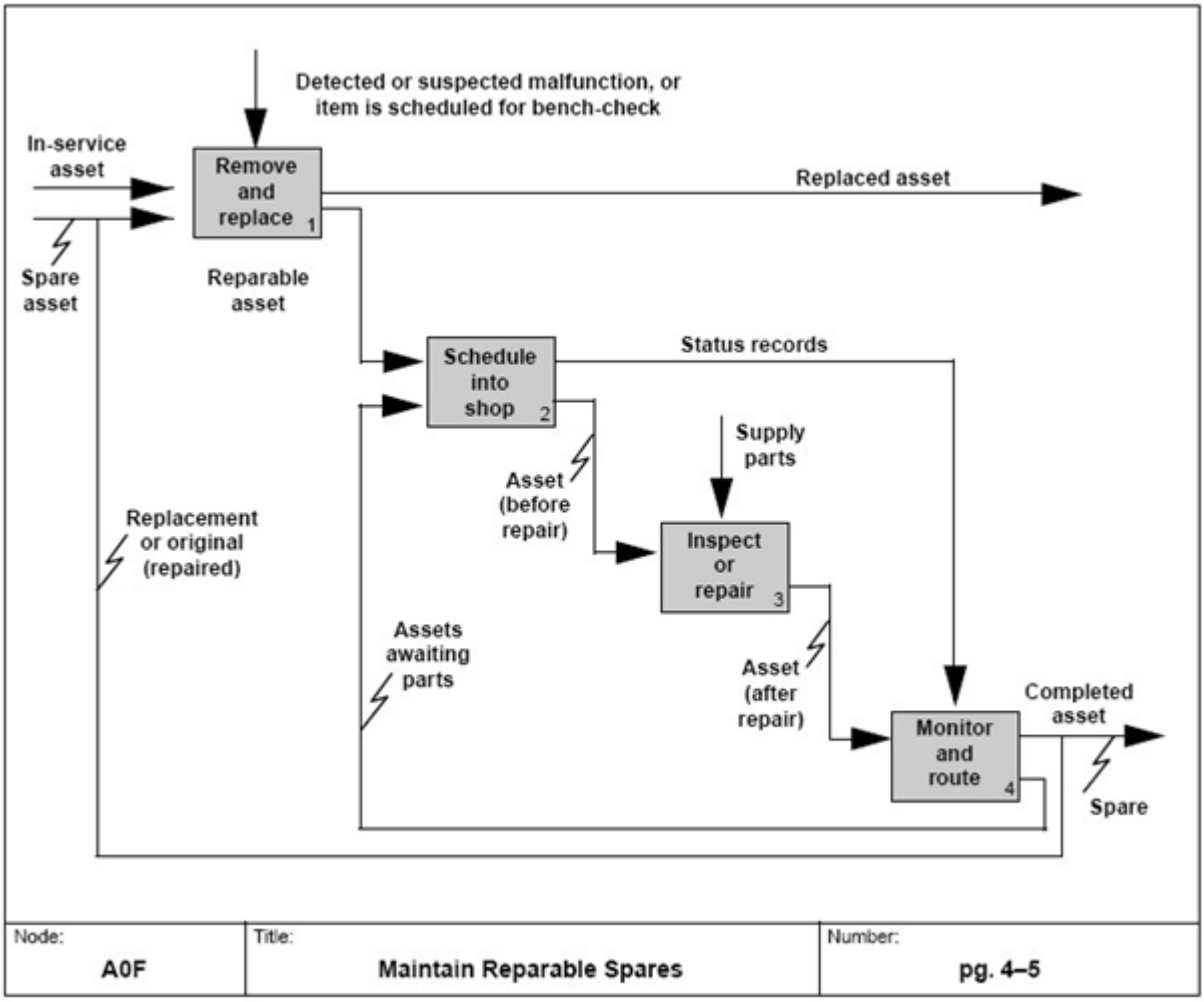


Рисунок 3.7 – Фрагмент функціональної моделі

Створення моделі «ЯК БУДЕ». Створення та впровадження корпоративної інформаційної системи призводить до зміни умов виконання окремих операцій, структури бізнес-процесів та підприємства в цілому. Це призводить до необхідності змінити системи бізнес-правил, що використовуються на підприємстві, модифікувати посадові інструкції працівників. Функціональна модель «ЯК БУДЕ» дозволяє виявити ці зміни вже на етапі проєктування майбутньої інформаційної системи. Застосування функціональної моделі «ЯК БУДЕ» дозволяє не тільки скоротити час впровадження інформаційної системи, але й знизити ризики, пов’язані з нечутливістю персоналу до інформаційних технологій.

Розподіл ресурсів. Функціональна модель дозволяє чітко визначити розподіл ресурсів між бізнес-операціями, що дозволяє оцінити ефективність використання ресурсів.

Це завдання особливо актуально при створенні нових бізнес-процесів на підприємстві. Наприклад, компанія, яка спеціалізується на розробці програмного забезпечення на замовлення, вирішила створити власний відділ продажів. Функціональна модель бізнес-процесу продажу програмного забезпечення дозволить керівництву компанії чітко визначити,

які ресурси необхідно виділити для забезпечення функціонування служби продажів, скільки співробітників необхідно залучити до нової служби, які функціональні обов'язки ці співробітники повинні виконувати тощо.

Побудова функціональної моделі системи (модель IDEF0).

Перш ніж визначити та описати порядок побудови моделі, потрібно зробити кілька зауважень:

1. Кожна модель SADT (Structured Analysis and Disign Tecchnique) вимагає точного визначення меж системи, цілей моделювання, перспективи, контексту системи;

2. Моделі SADT вимагають, щоб систему завжди розглядали з однієї точки зору, оскільки зміна точки зору може зробити модель неадекватною. Точка зору диктує вибір необхідної інформації та спосіб її подання;

3. Моделі SADT будуються з верхнього рівня «з голови». У них діаграмами нижнього рівня є деталізовані діаграми верхнього рівня. Кінцевим результатом є ієрархічна структура діаграм.

На початку побудови функціональної моделі потрібно чітко визначити:

- межі системи;
- мету моделювання;
- контекст системи;
- точку зору.

Тобто, потрібно дати відповідь на питання, які є джерелом будь-якого системного аналізу.

Побудова функціональної моделі полягає у функціональній декомпозиції системи. Виконуючи декомпозицію, окремі функції системи поділяються на підфункції і зображаються у вигляді ієрархічної структури. При аналізі систем, які існують, але не спроєктовані, найчастіше функціональна декомпозиція виконується на основі реальної структурної схеми системи. Перш за все визначають функції окремих підсистем і ставлять їх у відповідність з функціональними блоками. Функціональні блоки не обов'язково мають відповідати структурним підрозділам системи, оскільки кожен структурний підрозділ може виконувати кілька функцій або одну функцію можуть виконувати декілька підсистем. Дозволяється виконувати функціональну декомпозицію незалежно від структурної схеми системи та орієнтуватися лише на ті функції, які має виконувати кожна підсистема. Часто це робиться з метою реінжинірингу систем, тобто з метою реструктуризації та вдосконалення системи. Для систем, що проєктуються, функціональна декомпозиція виконується до розробки структурної схеми, а структурна схема розробляється вже на основі функціональної моделі.

Функціональна модель являє собою серію діаграм, а саме: контекстну діаграму, діаграми декомпозиції та деякі інші документи. Елементи контекстної діаграми та діаграми декомпозиції є такими:

- функціональні блоки (діяльності) – activity;
- дуги інтерфейсу (стрілки) – arrow.

Функціональні блоки діаграм діяльності зображені прямокутниками. Сторони прямокутника мають таке призначення:

– вхід (input);
– керування (control);
– вихід (output);
– механізм (mechanism).

Кожен блок має мати принаймні один елемент керування та одне вихідне значення. Зрозуміло, що блок, який не має початкового значення, не має бути на схемі (це робота, яка не дає жодного результату). Відсутність цих дуг при розборі діаграми буде вважатися помилкою. Дуги інтерфейсу (arrow) – стрілки на діаграмах декомпозиції встановлюють зв′язки між окремими функціональними блоками (activity). Слово «інтерфейс» в перекладі з англійської означає підключення, зв'язок, з'єднання. Тобто дуги з'єднують роботи між собою. Вони відповідають тим об'єктам, які циркулюють між блоками, передаються з блока в блок під час роботи системи. Усі дуги мають мати назви. При іменуванні дуги описуються об'єкти, яким вона відповідає. Назви дуг присвоюються на місцях встановлених міток. Мітки на дугах мають бути розміщені відповідно до таких правил:

– Кожна дуга має мати назву;
– Після розгалуження дузі можна дати нову назву або ні;
– Якщо після розгалуження дузі не дається нова назва, то вона містить усі об'єкти, які вона містила до розгалуження;
– Якщо після розгалуження дуга містить частину об'єктів, то їй необхідно дати ім'я та вказати (або очистити з назви) список об'єктів, яким вона відповідає;
– Після злиття дуга має бути перейменована.

На діаграмах декомпозиції можливе тунелювання дуг. Тунелювання – це умовне позначення дуг без нанесення їх на діаграму. Тунелювання має на меті спростити зображення діаграм і зробити їх легшими для читання. Дійсно, якщо діаграма декомпозиції має, наприклад, 6 функціональних блоків? і кожен блок має кілька дуг з різних сторін, то ці дуги можуть бути настільки переплетені, що прочитати діаграму буде важко. Тунелювання дозволяється лише для вторинних дуг. Дуги з батьківської діаграми, які застосовуються до всіх блоків у дочірній діаграмі, можуть не відображатися. Кінець такої дуги на виході з батьківської діаграми береться в дужки і не вказується в наступних діаграмах. Це позначення називається «не в дочірній діаграмі». Тунелювання також може бути таким, що дуга не з'являється на батьківській діаграмі, а з'являється на дочірній діаграмі. Така дуга позначається дужками на вході в блок, що означає «не на батьківській діаграмі».

Розглянемо процедуру побудови функціональної моделі. Використовуючи ці правила, ви можете побудувати функціональну модель як прямо на папері за допомогою олівця, так і за допомогою комп'ютера за допомогою програмного пакета BPwin. Побудувати діаграми на папері відносно

легко, коли вся модель має один або два рівні декомпозиції. Якщо таких рівнів більше, то є труднощі, які досить важко подолати. Тому функціональну модель рекомендується будувати за допомогою комп'ютера. Правила роботи з комп'ютером і введення моделі тут не будемо детально обговорюватися, а відзначимо основні моменти побудови. Зауважте, що перед початком роботи потрібно виділити місце на носіях, зокрема на жорсткому диску комп'ютера для розміщення моделі, тобто створити папку, а потім помістити всі результати в цю папку.

Після завантаження програмного пакета BPwin вам потрібно:

– відкрити новий документ;

– вибрати тип моделі IDEF0;

– дати їй власну назву;

– вказати інформацію про автора розробки;

– зберегти файл моделі в попередньо створеній папці.

Після цього приступайте до введення інформації відповідно до створеної моделі. Ця інформація має містити:

– визначення функцій системи;

– цілі моделювання;

– контекст огляду системи;

– точку зору, з якої розглядається система;

– відомості про авторів розробки;

– список джерел інформації.

Після побудови контекстної діаграми переходять до побудови наступних діаграм моделі. Ці діаграми є діаграмами розкладання. Діаграма розкладання будується в такому порядку. Якщо вибрати піктограму декомпозиції, діалогове вікно вказує тип діаграми, яку потрібно побудувати. Пакет BPwin дозволяє будувати складні моделі з різними типами діаграм. Ви маєте вказати діаграму IDEF0 і кількість блоків декомпозиції. Діалогове вікно для вибору кількості блоків декомпозиції за типом діаграми декомпозиції показано на рисунку 3.8.

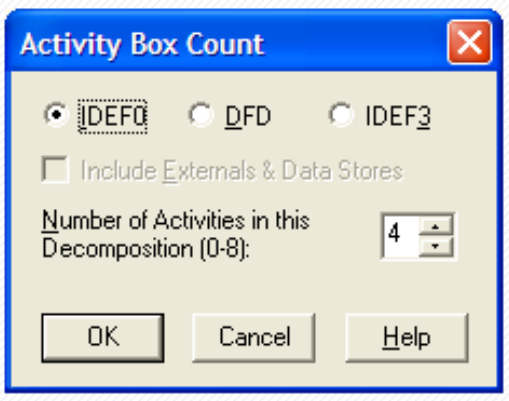


Рисунок 3.8 – Діалогове вікно вибору типу діаграми і кількості блоків декомпозиції

Рекомендується вибирати кількість блоків декомпозиції в діапазоні від 2 до 6. Обмеження пов’язані з тим, що в декомпозиції немає сенсу, якщо один блок замінюється іншим, а число 6 рекомендується, оскільки діаграма з більш ніж 6 блоками і зв’язками між ними стають складними і важко читаються (у пакеті BPwin можна ввести до 8 блоків). Після вибору типу діаграм і кількості блоків робіт з’являється форма діаграми декомпозиції, як показано на рисунку 3.9. Форма діаграми декомпозиції містить блоки робіт, розміщені зверху вниз, зліва направо. На цьому прикладі поясним зміст діаграм декомпозиції та правила їх побудови. Розміщення функціональних блоків зверху вниз, зліва направо називається домінуючим. У верхньому лівому кутку розташований блок роботи, яка вважається найбільш пріоритетною, нижче і праворуч – блоки менш пріоритетних функцій, або у випадку, коли на схемі показано післяопераційне виконання певних робіт, домінування блоків визначає порядок роботи. Блоки функціональної діаграми мають серійний номер і закреслений лінією верхній кут. Нумерація блоків відповідає їх пріоритету і служить для встановлення єдиного порядку блоків у всій моделі. Закреслений лівий верхній кут блока є ознакою того, що цей блок ще не зазнав операцій декомпозиції. Якщо ви пізніше розкладете цей блок, позначка буде видалена, якщо ні – вона залишиться на діаграмі.

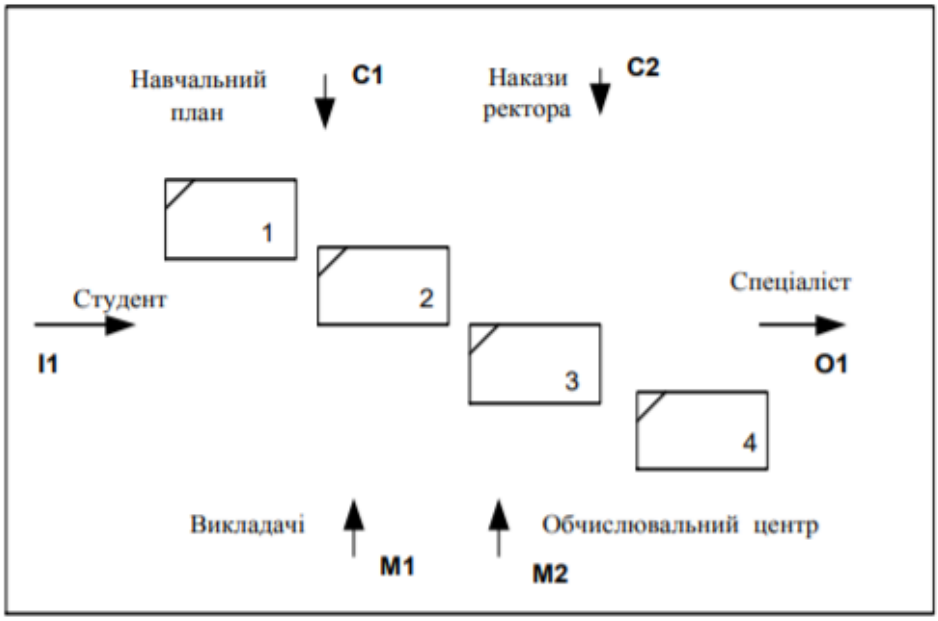


Рисунок 3.9 – Зразок бланка декомпозиції

Після побудови діаграми декомпозиції першого рівня починається декомпозиція її функціональних блоків. Декомпозиція виконується подібно до декомпозиції контекстної діаграми, з тією різницею, що перед декомпозицією потрібно вказати, який блок діаграми підлягає декомпозиції. Після декомпозиції діаграми першого рівня необхідно проаналізувати модель на наявність синтаксичних помилок, провести перевірку адекватності

моделі, а потім перейти до декомпозиції діаграм другого рівня. Те ж саме робиться після розкладання діаграм другого та наступних рівнів. Завершення декомпозиції визначається вимогами мети моделювання, а також тим, що блоки декомпозиції є елементарними роботами. Декомпозиції підлягають не всі роботи, а лише ті, які, виходячи з цілей моделювання, потребують декомпозиції. Після завершення декомпозиції моделі виходить ряд діаграм, приклад, показано на рисунку 3.10.

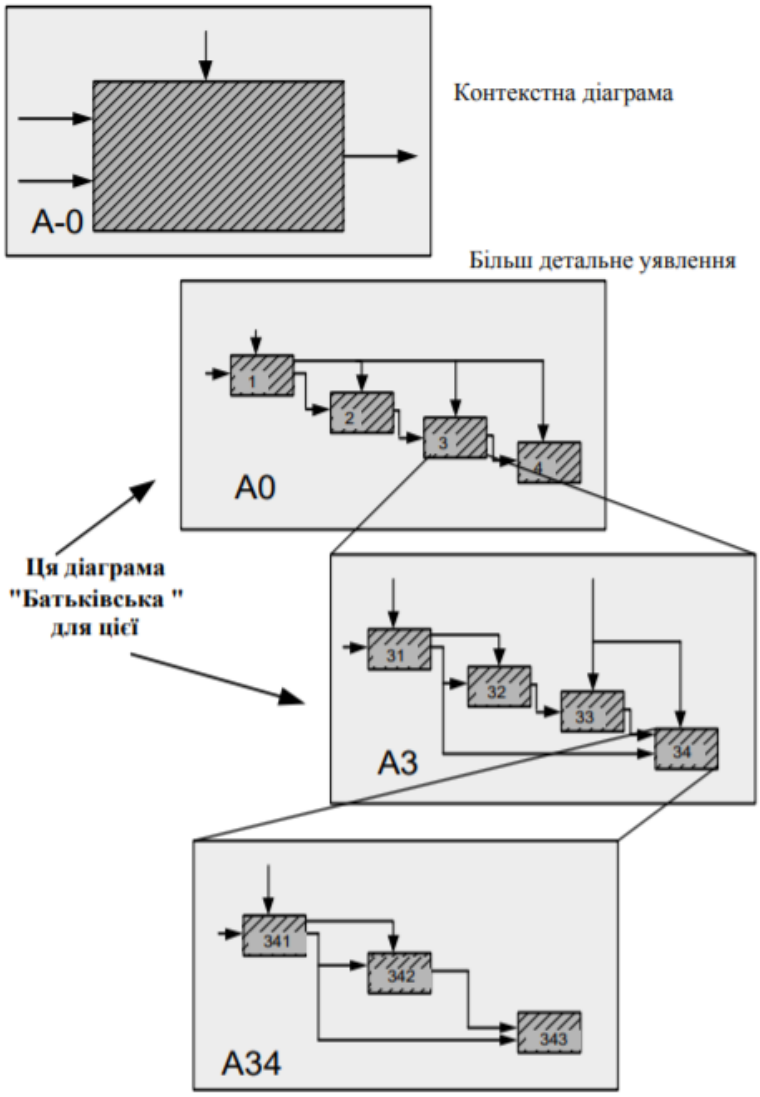


Рисунок 3.10 – Розміщення діаграм декомпозиції функціональної моделі

Кожна діаграма функціональної моделі системи має позначення. Система позначень така: контекстна діаграма позначається А-0, декомпозиція контекстної діаграми (діаграма першого рівня декомпозиції) – А0, розкладання функціональних блоків діаграми першого рівня – А1, А2, А3 ..., декомпозиція блоків діаграми А1 – двома цифрами А11, А12, А13 ..., діаграми А2 – двома цифрами А21, А22, А23…. Наступні діаграми розкладання позначаються літерою А з трьома цифрами, далі (якщо є) – чотирма циф-

рами тощо. Однак більшість моделей обмежені діаграмами рівня 3–4, оскільки дерево розкладання стає великим щодо глибини, а модель є складною. Для великих організаційних систем модель може містити сотні або тисячі аркушів. Якщо потрібно працювати з моделями, які мають багато діаграм на різних рівнях декомпозиції, їх поділяють на ряд менш складних моделей і розглядають незалежно. Такий поділ та об'єднання кількох моделей в одну забезпечується пакетом BPwin. Використовується це тоді, коли команді аналітиків потрібно працювати з великими системами. Це дозволяє кожному виконувати свою роботу самостійно, а потім поєднувати результати. Приклад функціональної моделі простої системи «Токарна дільниця» та « наведено на рисунках 3.11, 3.12. Модель складається з двох діаграм: контекстної та діаграми декомпозиції.

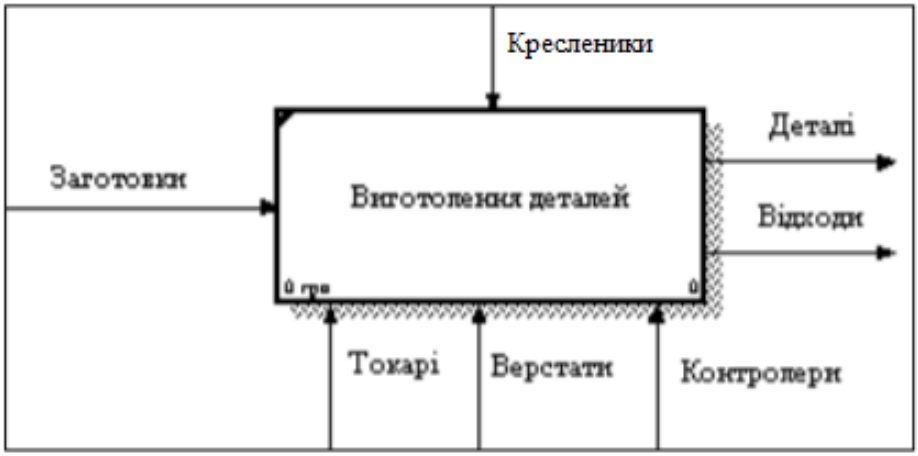


Рисунок 3.11 – Контекстна діаграма системи «Токарна дільниця»

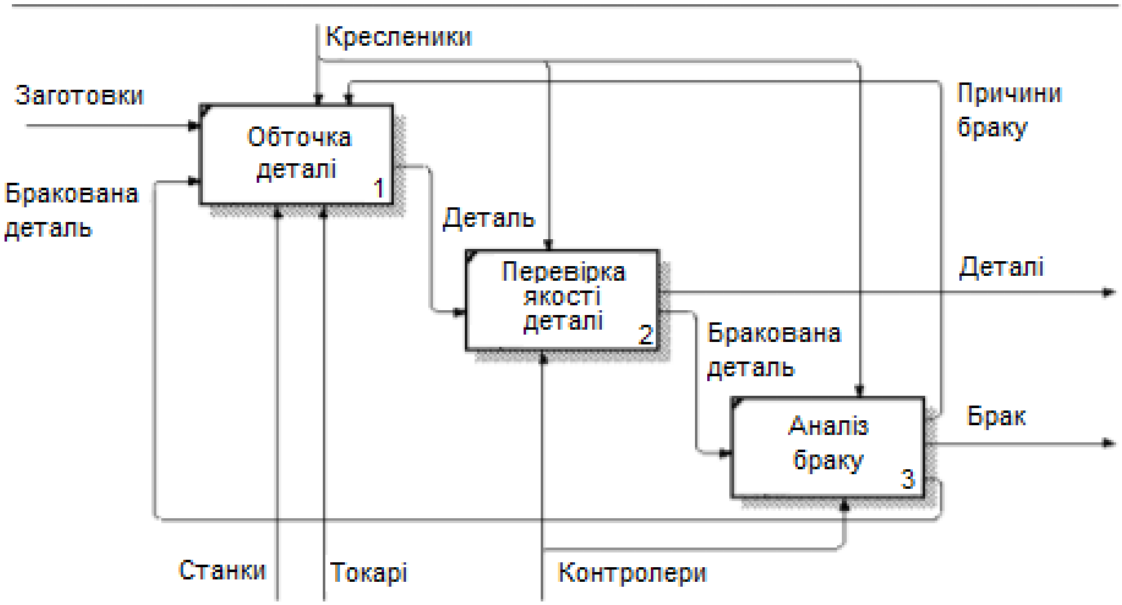


Рисунок 3.12 – Діаграма декомпозиції системи «Токарна дільниця»

Діаграма декомпозиції складається з трьох робіт, а саме: «Обточування деталі», «Перевірка якості деталі» та «Аналіз браку». Пріоритетною і першою за чергою є робота «Обточування деталі». Токарі, згідно з кресленниками, обточують заготовки на верстатах і отримують деталь. Виготовлені деталі відправляються на перевірку якості. Перевірку проводять контролери за кресленниками. Перевірені деталі, залежно від результату перевірки, передаються в інші системи (є вихідними значеннями), або бракуються і передаються для аналізу причин браку. Після аналізу причин браку, проведеного контролерами, одні деталі повертаються на обточування, а інші повністю бракуються і залишають систему як непоправний брак. Після аналізу причин браку інформація про результати аналізу передається токарям, які обточують деталі. Ця інформація служить для них орієнтиром і має на меті покращити продуктивність.

3.2 Моделювання процесів (IDEF3)

Основою діяльності будь-якої організації є її бізнес-процеси, які визначаються цілями і завданнями діяльності суб'єкта господарювання. Процеси забезпечують здійснення всіх видів діяльності підприємства, пов'язаних з виробництвом товарів та/або послуг. Для кожного виду робіт, що входять до загального процесу господарської діяльності, визначені тимчасові характеристики, які визначають його місце в загальній послідовності робіт, умови ініціації та час виконання.

Використовується декілька різних методів, заснованих як на структурному, так і на об'єктно-орієнтованому підходах до моделювання.

Серед найбільш поширених методів можна виділити:

- метод функціонального моделювання SADT (IDEF0);
- метод моделювання процесу IDEF3;
- моделювання потоків даних DFD;
- метод ARIS;
- Метод Еріксона-Пенкера [2,3].

Сьогодні найвідомішими мовами (позначеннями) графічного моделювання бізнес-процесів є UML, ARIS, IDEF (IDEF0, IDEF3 в програмній інтерпретації BPwin), BPMN.

В даний час на ринку комп'ютерних технологій існує ряд спеціальних програм для моделювання бізнес-процесів, які дозволяють проводити обстеження підприємства та будувати моделі. Це такі програмні засоби, як Ramus, Business Studio, Fox Manager, BPwin та інші [4]. Розглянемо аналіз та моделювання бізнес-процесів, які проводилися на базі одного з цехів підприємства – А-5 з виробництва аміаку. Для відображення функціонування діяльності використовувалося програмне забезпечення Ramus – кро-

сплатформна система моделювання та аналізу бізнес-процесів. У роботі за допомогою Рамуса побудована загальна контекстна діаграма діяльності цеху А-5, яка зображена на рис. 3.13.

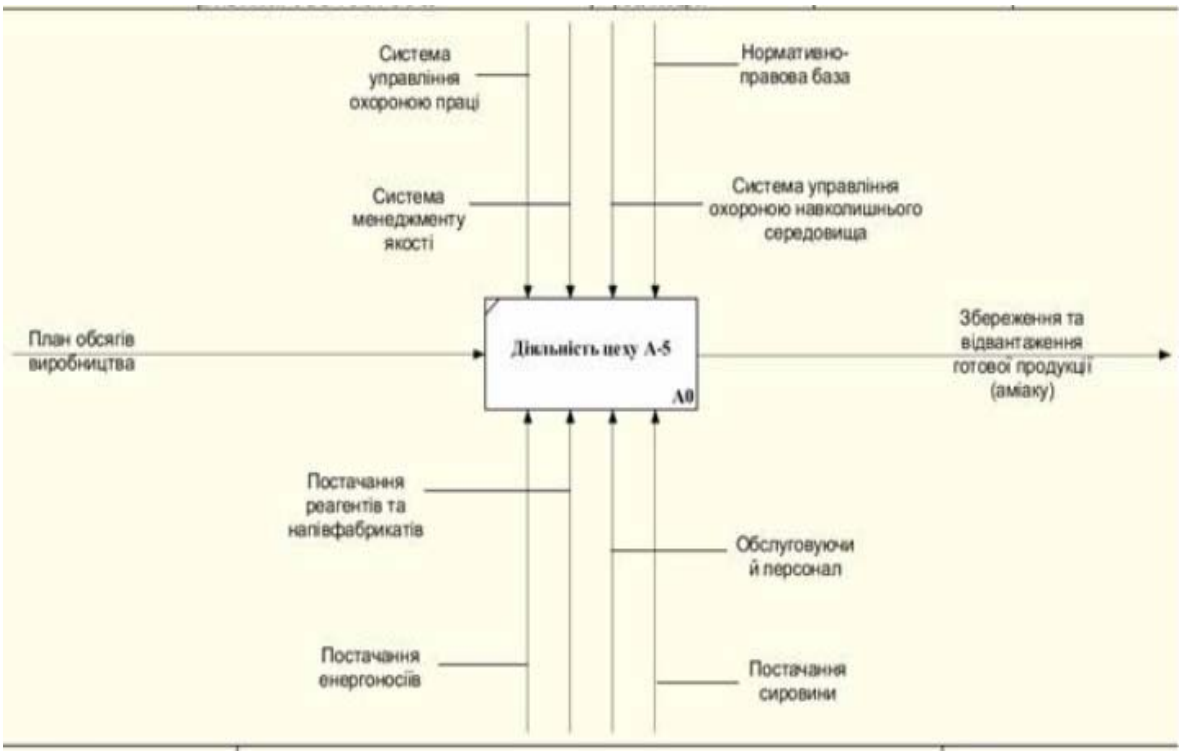


Рисунок 3.13 – Контекстна діаграма діяльності цеху А-5
за методологією IDEF0

На рисунку 3.13 показано, що діяльність цеху можна представити за допомогою контекстної діаграми, яка показує вхідну інформацію (бізнес-процеси), механізми, вплив управління та вихідну інформацію. На вхід контекстної діаграми моделювання бізнес-процесів цеху А-5 подаються планові завдання (обсяги виробництва), які встановлюються для цеху керівництвом підприємства.

До механізмів впливу та ресурсів належать: постачання сировини, реагентів, напівфабрикатів та енергоносіїв, необхідних для виробництва аміаку; обслуговувальний персонал.

Керувальний плив охоплює: нормативно-правову базу; систему управління якістю; охорону праці та систему екологічного менеджменту. Вихід моделі відображає бізнес-процеси зберігання та відвантаження готової продукції (аміаку).

На рис. 3.14 наведено детальний процес виробництва аміаку у вигляді моделі бізнес-процесу за методологією IDEF3.

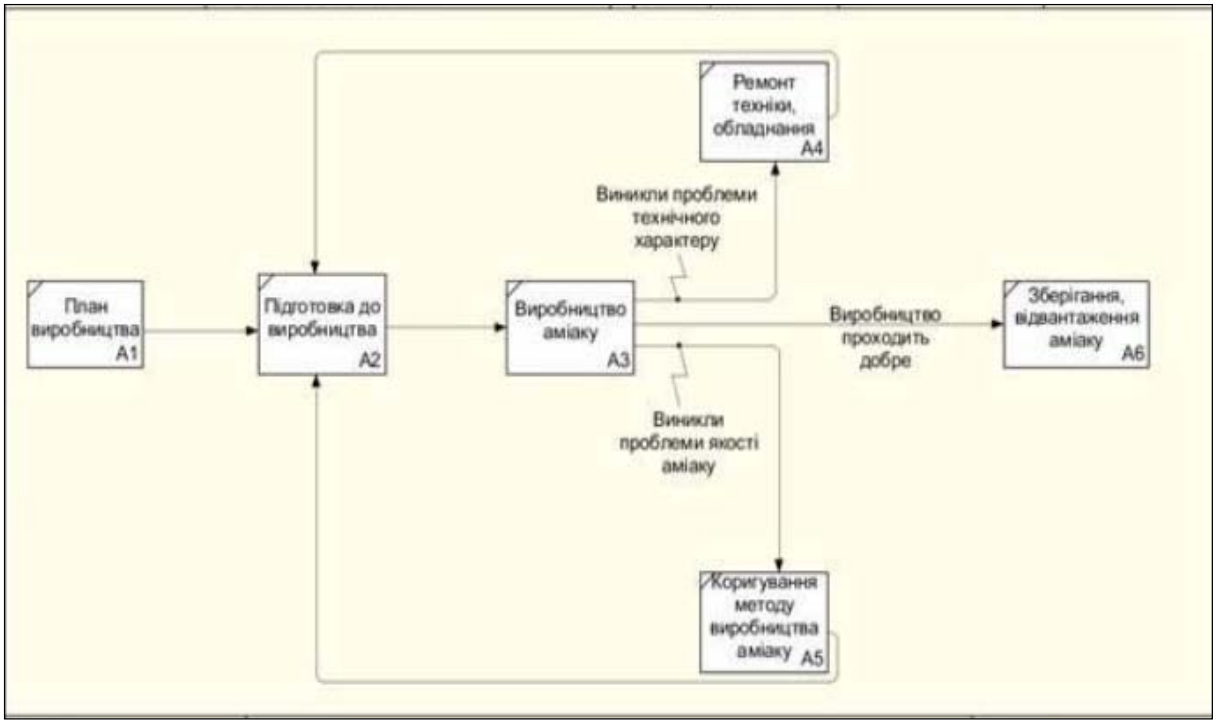


Рисунок 3.14 – Моделі бізнес-процесів виробництва аміаку
за методологією IDEF3

З рис. 3.14 видно, що модель бізнес-процесу за методологією IDEF3, яка описує етапи технологічного циклу виробництва аміаку, складається з кількох кроків:

1. Першим кроком є отримання плану виробництва;
2. Другий крок – підготовка до виробництва, що охоплює постачання матеріалів і ресурсів.
3. Третій крок – це безпосередньо технологічний процес виробництва аміаку, який може відбуватися по-різному:
   - під час виробництва можуть виникнути технічні проблеми, в цьому випадку пошкодження усуваються, і підприємство повертається на другий етап;
   - проблеми якості продукції – відбувається аналіз методів виробництва та їх зміна, в такому випадку технологічний цикл теж повертається на другий етап;
   - якщо технологічний цикл виробництва аміаку проходить без збоїв, технологічний цикл закінчується четвертою стадією – зберіганням або відвантаженням аміаку.

Таким чином, моделювання бізнес-процесів для цеху А-5 дає змогу знайти шляхи підвищення якості та швидкості виробництва при зниженні витрат, підвищення професіоналізму працівників та підвищення конкурентоспроможності продукції.

Під моделлю бізнес-процесу будемо розуміти його формалізований (графічний, табличний, текстовий, символічний) опис, що відображає реальну або ймовірну діяльність організації. Модель бізнес-процесу зазвичай містить такі дані:

– набір етапів процесу;
– бізнес-функції;
– порядок виконання бізнес-функцій;
– механізми контролю та управління в рамках бізнес-процесу;
– виконавці кожної бізнес-функції;
– вхідні документи (інформація), вихідні документи (інформація);
– ресурси, необхідні для виконання кожної бізнес-функції;
– документація (умови), що регламентують виконання кожної бізнес-функції;
– параметри, що характеризують виконання бізнес-функцій і процесу в цілому.

Методи моделювання бізнес-процесів можна розділити за двома основними принципами: структурні методи та об'єктно-орієнтовані. На рис. 3.15 наведена класифікація методів моделювання бізнес-процесів.

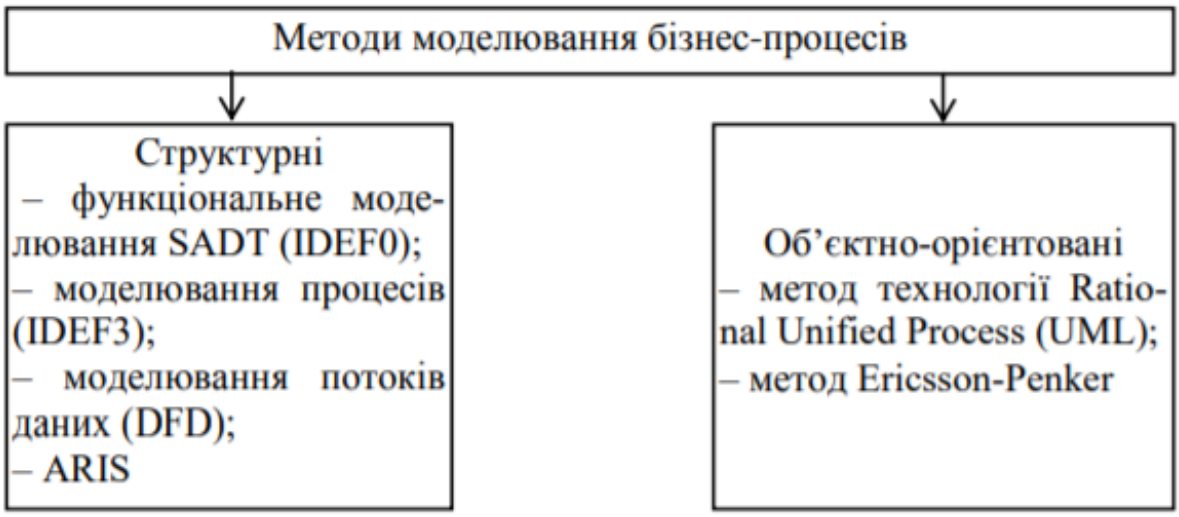


Рисунок 3.15 – Класифікація методів моделювання бізнес-процесів

Метод SADT (Structured Analysis and Design Technique) є методом процесного підходу до управління і може використовуватися для моделювання різних бізнес-процесів і систем.

Метод IDEF3 призначений для моделювання послідовності дій і взаємозалежностей між ними в межах процесів.

Метод DFD (Data Flow Diagrams) дозволяє продемонструвати, як кожен процес перетворює свої вхідні дані у вихідні, а також визначити зв'язки між цими процесами.

Методологія моделювання ARIS (Architecture of Integrated Information System) базується на теорії побудови інтегрованих інформаційних систем. ARIS підтримує чотири типи моделей:

- організаційні;
- функціональні;
- інформаційні;
- моделі керування.

З наведеного аналізу методів моделювання бізнес-процесів можна зробити висновки, що доцільно використовувати методологію IDEF0 та IDEF3 для побудови концептуальних моделей бізнес-процесів функціонування веб-порталів управління. За допомогою IDEF0 модель буде побудована на найбільш абстрактному рівні, подальша деталізація буде виконана у вигляді IDEF3-діаграм.

Розглянемо побудову концептуальної моделі на прикладі бізнес-процесу «Ліцензування діяльності з надання освітніх послуг у сфері вищої освіти». На рис. 3.16 наведена IDEF0-діаграма.

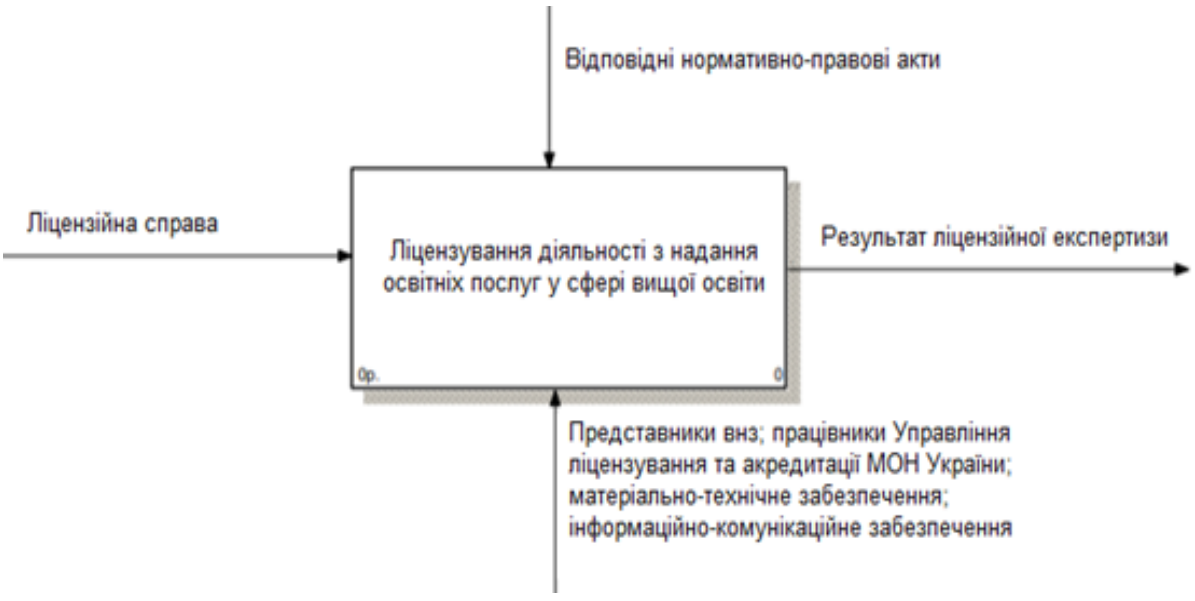


Рисунок 3.16 – Функціональна діаграма бізнес-процесу

Деталізація функціонального блока показана на рис. 3.17 у вигляді діаграми IDEF3.

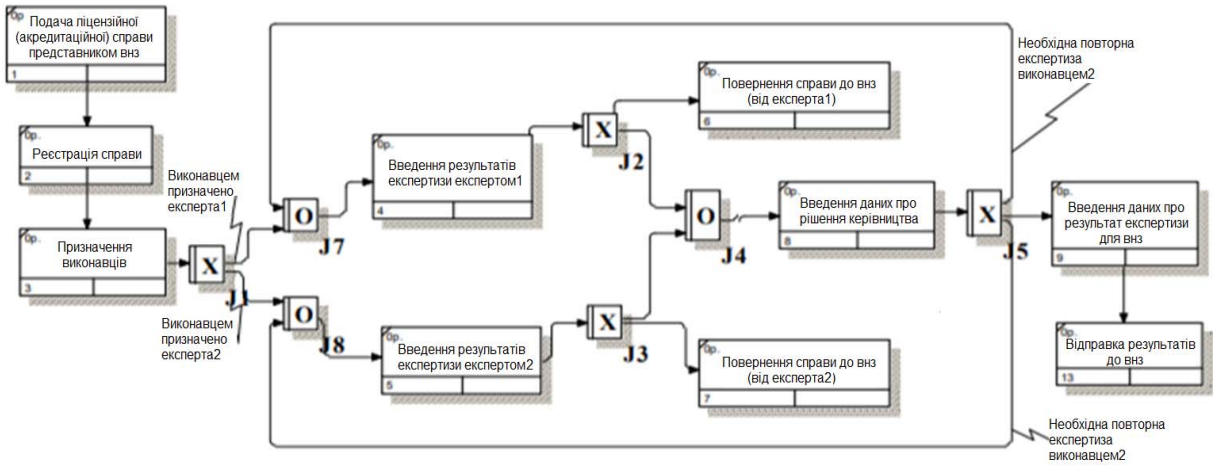


Рисунок 3.17 – IDEF3-діаграма бізнес-процесу «Ліцензування діяльності з надання освітніх послуг у сфері вищої освіти»

Як видно з рис. 3.17, діаграма IDEF3 містить набір бізнес-функцій (дій або етапів бізнес-процесу), зв'язків, посилань і перехрещень [2]. Перехрещення використовуються для відображення набору дій, які можна або потрібно виконати до початку наступної дії. Розрізняють перехрещення для злиття (Fan-in Junction) і розгалуження (Fan-out Junction) стрілок. Перехрещення не можна одночасно використовувати як для злиття, так і для розгалуження. Існує кілька типів перехрещень, зміст кожного з них наведено в таблиці 3.1.

Таблиця 3.1. – Типи перехрещень

Найменування	Зміст у випадку злиття стрілок (Fan-in Junction)	Зміст у випадку розгалуження стрілок (Fan-out Junction)
Asynchronous AND	Всі попередні процеси мають бути завершені	Всі наступні процеси мають бути запущені
Synchronous AND	Всі попередні процеси завершені одночасно	Всі наступні процеси запускаються одночасно
Asynchronous OR	Один або кілька попередніх процесів мають бути завершені	Один або кілька наступних процесів мають бути запущені
Synchronous OR	Один або кілька попередніх процесів завершені одночасно	Один або кілька наступних процесів запускаються одночасно
XOR (Exclusive OR)	Тільки один попередній процес завершений	Тільки один наступний процес запускається

Якщо на кожній діаграмі IDEF3 в межах кожної дії бізнес-процесу виконуються інші дії, потрібно виконати їх подальшу декомпозицію у вигляді дочірніх діаграм IDEF3. Для наведеного бізнес-процесу не потрібно додаткової декомпозиції.

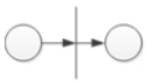
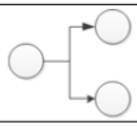
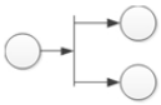
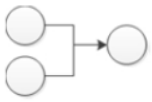
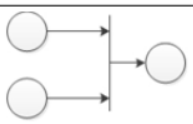
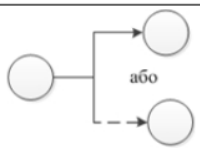
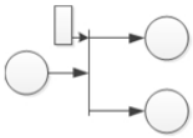
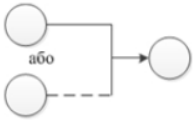
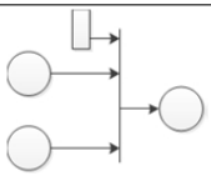
Формалізований опис бізнес-процесів.

Концептуальні моделі надають найкращий спосіб подання логіки бізнес-процесу, але їх не можна використовувати для управління процесами. Тому виникає необхідність трансформації концептуальних моделей у формалізований опис, який у майбутньому можна використовувати для управління роботою веб-порталу.

В Е-мережах управління – Control E-net (CEN) – базовий набір переходів звичайних Е-мереж розширено введенням переходів-черг, які виконують функції розміщення міток у черзі з різною дисципліною обслуговування.

Базовий набір переходів CEN, крім додаткових переходів-черг, реалізує п’ять основних типів взаємодії паралельно-послідовних процесів. Саме за допомогою цих переходів можуть бути формально визначені особливості концептуальної моделі мережного графіка, як це показано в табл. 3.2.

Таблиця 3.2 – Базисний набір переходів CEN

Вузол плану робіт	Е-мережний вузол	Семантика
		Т-перехід («виконання роботи»). Перехід спрацьовує за наявності мітки у вхідній позиції і відсутності її у вихідний позиції
		F-перехід («розгалуження шляхів»). Спрацьовує за наявності мітки у вхідній позиції і відсутності міток у вихідних позиціях
		J-перехід («злиття шляхів»). Перехід спрацьовує при наявності міток в обох вхідних позиціях і відсутності у вихідній
		X-перехід («перемикач між альтернативними шляхами»). Напрямок розвитку процесу визначається значення вирішальної процедури R.
		Y-перехід («вибір»). Y-перехід здатний відображати умови продовження процесу у випадку злиття декількох шляхів. Корисний для відображення зворотних зв’язків, які можуть допускатися у плані

3.3 Моделювання потоків даних (DFD)

Діаграма потоків даних (Data Flow Diagram) – модель проєктування, графічне подання «потоків» даних в інформаційній системі. Діаграму потоку даних також можна використовувати для візуалізації процесів обробки даних (структурне проєктування).

Вважається звичайним для розробника спочатку нарисувати діаграму потоку даних на контекстному рівні, щоб показати, як система взаємодіє з зовнішніми модулями. В подальшому ця діаграма додатково уточнюється

шляхом деталізації процесів і потоків даних, щоб показати розширену систему, що розробляється.

Діаграми потоку даних містять чотири типи графічних елементів:

– процеси – це перетворення даних у межах описаної системи;
– сховища даних (репозиторії);
– зовнішні щодо системи сутності;
– потоки даних між елементами трьох попередніх типів.

Інформаційна система отримує зовнішні потоки даних. Поняття зовнішньої сутності використовується для позначення елементів середовища функціонування системи. Усередині системи відбуваються процеси перетворення інформації, які породжують нові потоки даних. Потоки даних можуть надходити на вхід до інших процесів, міститися у сховищах даних, передаватись зовнішнім об'єктам.

Модель DFD, як і більшість інших структурних моделей, є ієрархічною моделлю. Кожен процес можна розкласти, тобто розбити на структурні компоненти, взаємозв'язок між якими в однакових позначеннях можна показати на окремій схемі. При досягненні необхідної глибини декомпозиції процес нижнього рівня супроводжується міні-специфікацією (текстовим описом).

Метою методології потоку даних є побудова моделі системи у вигляді діаграми потоку даних (Data Flow Diagram – DFD), яка забезпечує правильний опис виходів (відповідей системи у вигляді даних) з заданим впливом на вхід системи (подані сигнали через зовнішні інтерфейси). Діаграми потоків даних є основним засобом моделювання функціональних вимог до проєктованої системи.

Для створення діаграми потоку даних використовуються такі основні концепції:

– потоки даних, які є абстракціями, що використовуються для моделювання передачі інформації (або фізичних компонентів) від однієї частини системи до іншої. Потоки на діаграмах зображені іменованими стрілками, орієнтація яких вказує напрямок руху інформації;

– процеси (роботи) перетворення вхідних потоків даних у вихідні, які полягають у виробництві вихідних потоків із вхідних даних відповідно до дії, визначеної назвою процесу. Назва процесу має містити дієслово в неозначеній формі з таким доповненням (наприклад, «отримувати документи на відвантаження продукції»). Кожен процес має унікальний номер для посилання в діаграмі, який можна використовувати разом з номером діаграми для отримання унікального індексу процесу в усій моделі.

Основою даної методології є побудова моделі аналізованої ІС – проєктованої або реально існуючої. Відповідно до методології модель системи визначається як ієрархія діаграм потоків даних (ДПД або DFD), що описують асинхронний процес перетворення інформації від її надходження в систему до доставки користувачеві. Діаграми верхніх рівнів ієрархії (контекстні діаграми) визначають основні процеси або підсистеми ІС із зо-

внішніми входами та виходами. Вони детально описані за допомогою діаграм нижнього рівня. Ця декомпозиція продовжується, створюючи багаторівневу ієрархію діаграм, доки не буде досягнуто такого рівня декомпозиції, на якому процеси стають елементарними і деталізувати їх далі неможливо.

Джерела інформації (зовнішні сутності) генерують інформаційні потоки (потоки даних), які передають інформацію до підсистем або процесів. Вони, у свою чергу, перетворюють інформацію та генерують нові потоки, які передають інформацію іншим процесам чи підсистемам, накопичувачам даних чи зовнішнім сутностям – споживачам інформації. Таким чином, основними компонентами діаграм потоків даних (рис. 3.18) є:

- зовнішні суб'єкти;
- системи / підсистеми;
- процеси;
- накопичувачі даних;
- потоки даних.

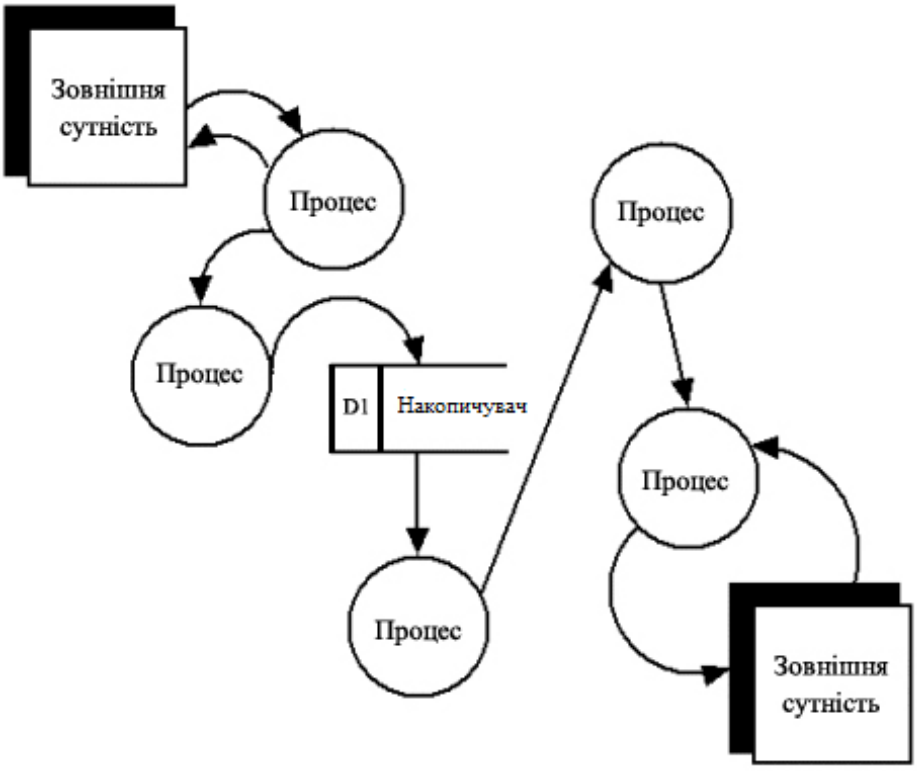


Рисунок 3.18 – Приклад ДПД

Сховище (накопичувач) даних дозволяє визначити дані, які будуть зберігатися в пам'яті між процесами. Насправді сховище являє собою «зрізи» потоків даних у часі. Інформацію, яку сховище містить, можна використовувати в будь-який момент після її отримання, а дані можна вибирати в будь-якому порядку. Назва сховища має визначати його вміст і бути іменником.

Зовнішня сутність – це матеріальний об’єкт або особа, яка виступає джерелом або одержувачем інформації, наприклад, клієнти, персонал, постачальники, замовники, склад тощо. Визначення об’єкта або системи як зовнішньої сутності вказує на те, що вони знаходяться за межами аналізованої ІС. У процесі аналізу деякі зовнішні сутності можуть бути перенесені всередину діаграми аналізованої ІС, якщо необхідно, або, навпаки, частина процесів ІС може бути вилучена з діаграми та подана як зовнішні сутності.

Зовнішню сутність позначають прямокутником (рис. 3.19), який розташований над діаграмою і відкидає на неї тінь, щоб виділити цей символ серед інших символів.

Рисунок 3.19 – Зовнішня сутність

Системи та підсистеми – при побудові моделі складної ІС вона може бути подана в найбільш загальному вигляді на так званій діаграмі контексту як єдина система в цілому, а може бути розкладена на ряд підсистем.

Підсистема (або система) на контекстній діаграмі зображена на рис. 3.20.

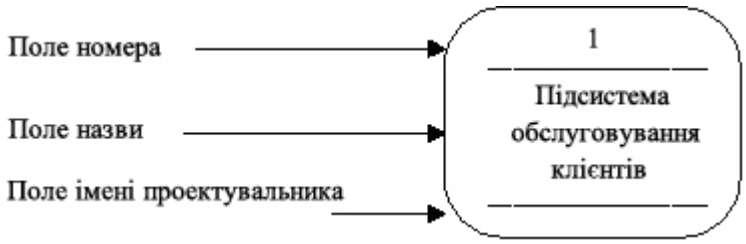


Рисунок 3.20 – Підсистема

Для ідентифікації використовується номер підсистеми. Назва підсистеми вноситься в поле імені у вигляді речення з підметом і відповідними означеннями та доповненнями.

Процес являє собою перетворення вхідних потоків даних у вихідні за певним алгоритмом. Фізично процес може бути реалізований різними способами: це може бути підрозділ організації (відділу), який виконує обробку вхідних документів і звітів, програма, апаратно реалізований логічний пристрій тощо.

Процес на схемі потоку даних показано на рис. 3.21.

Рисунок 3.21 – Процес: а) нотація процесу за Gane and Sarson;
б) Нотація процесу за Yourdon and Coad

Для ідентифікації використовується номер процесу. У поле назви введіть назву процесу у вигляді речення з активним однозначним дієсловом у неозначеній формі (обчислити, обчислити, перевірити, визначити, створити, отримати), а потім іменники в знахідному відмінку:

– «Ввести інформацію про клієнта»;

– «Видати інформацію про поточні витрати»;

– «Перевірити кредитоспроможність клієнта».

Нагромаджувач (накопичувач) даних може бути фізично реалізований у вигляді шухляди у картотеці, таблиці в ОЗП, файлу на магнітних носіях тощо. Нагромаджувач даних на схемі потоку даних показано на рисунку 3.22.

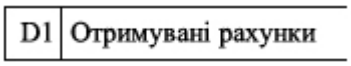


Рисунок 3.22 – Нагромаджувач даних

Нагромаджувач даних позначається буквою «D» і будь-яким числом. Назва нагромаджувача обрана з міркувань найбільшої інформативності для проєктувальника.

Потік даних визначає інформацію, яка передається через з’єднання від джерела до приймача. Фактичним потоком даних може бути інформація, що передається по кабелю між двома пристроями, електронна пошта, магнітні стрічки або дискети, які передаються з одного комп’ютера на інший тощо.

Потік даних на діаграмі зображено лінією, що закінчується стрілкою, що показує напрямок потоку (рис. 3.23). Кожен потік даних має назву, яка відображає його зміст.

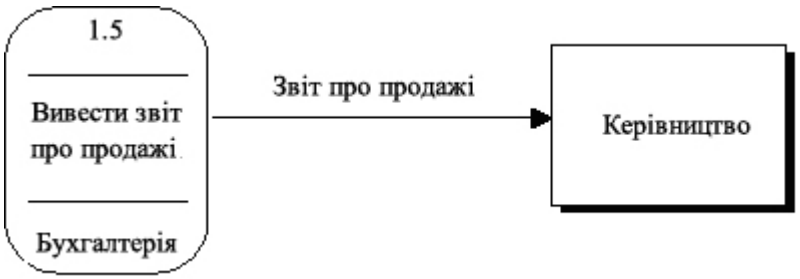


Рисунок 3.23 – Потік даних

Побудова ієрархії діаграм потоків даних. Першим кроком у побудові ієрархії діаграм потоків даних є побудова контекстних діаграм. Зазвичай при проєктуванні простих інформаційних систем будується єдина контекстна діаграма з топологією у формі зірки, в центрі якої знаходиться так званий основний процес, який підключається до приймачів і джерел інформації, за допомогою яких з системою взаємодіють користувачі та інші зовнішні системи.

Якщо складна система обмежена однією контекстною діаграмою, вона буде містити дуже велику кількість джерел і приймачів інформації, які важко розмістити на аркуші паперу нормального формату, і, крім того, єдиний основний процес не розкриває структури розподіленої системи. Ознаками складності (у сенсі контексту) можуть бути:

– наявність великої кількості зовнішніх сутностей (десять і більше);

– розподілений характер системи;

– багатофункціональність системи з групою функцій, які вже сформовані або ідентифіковані в окремих підсистемах.

Для складних інформаційних систем будується ієрархія діаграм контексту. Контекстна діаграма верхнього рівня містить не один основний процес, а набір підсистем, сполучених потоками даних. Контекстні діаграми наступного рівня детально описують контекст і структуру підсистем.

Розробка контекстних діаграм вирішує проблему суворого визначення функціональної структури інформаційної системи на першому етапі її проєктування, що особливо важливо для складних багатофункціональних систем.

Для кожної підсистеми, поданої в контекстних діаграмах, її деталізація виконується за допомогою діаграм потоків даних. Кожен процес діаграми потоків даних, у свою чергу, можна деталізувати за допомогою іншої діаграми потоків даних або міні-специфікації. При деталізації необхідно дотримуватися таких правил:

– правило *балансування* – означає, що при деталізації підсистеми або процесу, деталізувальна діаграма у ролі зовнішніх джерел/приймачів даних може мати лише ті компоненти (підсистеми, процеси, зовнішні сутності, накопичувачі даних), з якими підсистема або процес має інформаційний зв’язок на батьківській діаграмі;

– правило *нумерації* – означає, що при деталізації процесів має зберігатися їх ієрархічна нумерація. Наприклад, процеси, що деталізують процес номер 8, отримують номери 8.1, 8.2, 8.3 тощо.

Міні-специфікація є найвищою вершиною ієрархії діаграм потоків даних. Рішення про деталізацію процесу та використання міні-специфікації приймає аналітик на основі таких критеріїв:

– наявність у процесі відносно невеликої кількості вхідних і вихідних потоків даних (2–3 потоки);

– можливість описувати перетворення даних процесом у вигляді послідовного алгоритму;

– виконання процесом єдиної логічної функції перетворення вхідної інформації у вихідну;

– можливість описати логіку процесу за допомогою міні-специфікації невеликого обсягу (не більше 20–30 рядків).

При побудові ієрархії діаграм потоків даних необхідно переходити до деталізації процесів лише після визначення вмісту всіх потоків і накопичувачів даних, що описується за допомогою структур даних. Структури даних будуються з елементів даних і можуть містити альтернативи, умовні входження та ітерації. Умовний запис означає, що цей компонент може бути відсутнім у структурі. Альтернатива означає, що структура може містити один з цих елементів. Ітерація означає введення будь-якої кількості елементів у зазначеному діапазоні. Для кожного елемента даних можна вказати його тип (неперервні або дискретні дані). Для неперервних даних можуть бути вказані: одиниця виміру (кг, см тощо), діапазон значень, точність подання та форма фізичного кодування. Для дискретних даних можна вказати таблицю дійсних значень.

Після побудови повної моделі системи її необхідно перевірити (перевірити на повноту та узгодженість). У повній моделі всі її об’єкти (підсистеми, процеси, потоки даних) мають бути детально описані. Виявлені об’єкти, які не є деталізованими, потрібно деталізувати, повернувшись до попередніх кроків розробки. У послідовній моделі для всіх потоків даних і накопичувачів даних необхідно дотримуватися правил зберігання інформації: всі дані, які кудись надходять, мають бути прочитані, а всі прочитані дані мають бути записані.

Ще раз зауважимо, що зовнішня сутність – це матеріальний об’єкт поза контекстом системи, який є джерелом або приймачем системних даних. Її ім’я має містити іменник, наприклад «склад запасних частин». Передбачається, що об’єкти, подані як зовнішні об’єкти, не мають брати участі в жодній обробці.

Процес побудови діаграми потоку даних (DFD), як зазначалося вище, починається зі створення так званої базової діаграми типу «зірка», яка показує процес моделювання та всі зовнішні сутності, з якими він взаємодіє (рис. 3.24).

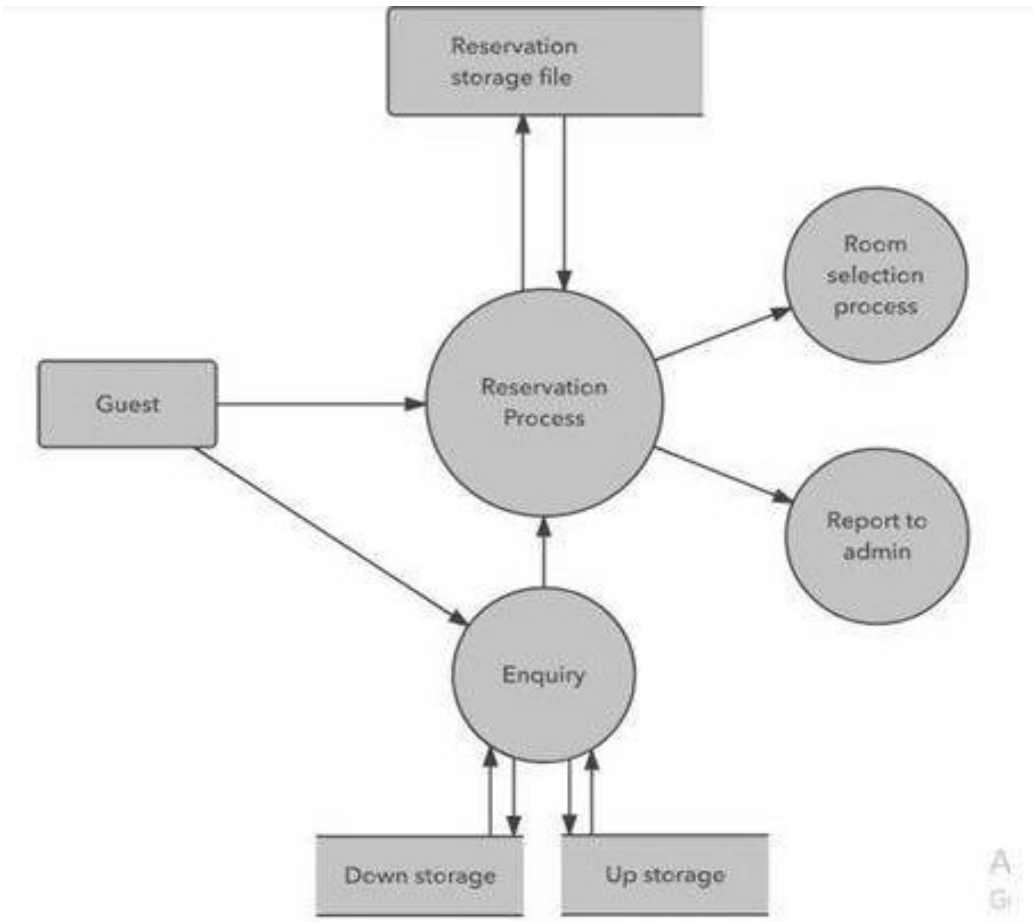


Рисунок 3.24 – Діаграма типу «зірка»

У разі складного основного процесу він відразу подається у вигляді декомпозиції на ряд взаємодійних процесів.

Критерії складності методології потоку даних:

– наявність великої кількості зовнішніх суб'єктів;

– багатофункціональність системи;

– його розподілений характер.

Повнота діаграми забезпечується, якщо в системі немає «завислих» процесів, які не використовуються в процесі перетворення вхідних потоків у вихідні.

Узгодженість системи забезпечується реалізацією наборів формальних правил щодо можливих типів процесів:

– на діаграмі не може бути потоку, який з'єднує дві зовнішні сутності – ця взаємодія вилучається з розгляду;

– жоден суб'єкт не може безпосередньо отримувати або надсилати інформацію в сховище даних – сховище даних є пасивним елементом, керованим процесом інтерфейсу;

– два сховища не можуть безпосередньо обмінюватися інформацією – ці сховища необхідно об'єднати.

До **переваг** техніки DFD можна віднести:

– здатність однозначно ідентифікувати зовнішні сутності, аналізуючи потік інформації всередині та поза системою;

– можливість проєктування зверху вниз, що полегшує побудову моделі «як має бути»;

– наявність специфікацій процесу нижчого рівня, що дозволяє подолати логічну незавершеність функціональної моделі та побудувати повну функціональну специфікацію системи розробки.

До **недоліків** техніки DFD можна віднести:

– необхідність штучного запровадження процесів управління, оскільки керувальні ефекти (потоки) та процеси управління в умовах DFD нічим не відрізняються від звичайних;

– відсутність поняття часу, тобто відсутність аналізу часових інтервалів при перетворенні даних (всі тимчасові обмеження мають бути введені в специфікації процесу).

3.4 Методологія моделювання бізнес-процесів ARIS

Для моделювання бізнес-процесів використовується декілька різних методів, які базуються як на структурному, так і на об’єктно-орієнтованому підходах до моделювання. Однак класифікація самих методів на структурні та об’єктні є досить умовною, оскільки найбільш розроблені методи використовують елементи обох підходів. Давайте коротко розглянемо характеристики найбільш поширених методів:

– метод функціонального моделювання SADT (IDEF0);

– метод моделювання процесу IDEF3;

– моделювання потоків даних DFD;

– метод ARIS;

– метод Еріксона-Пенкера (Ericsson-Penker);

– технологія раціонального уніфікованого процесу – Rational Unified Process.

Метод SADT (Structured Analysis and Design Technique) вважається класичним методом управління на основі процесів, основним принципом якого є структурування організації відповідно до її бізнес-процесів. Бізнес-модель відповідає таким вимогам:

– верхній рівень моделі відображає лише контекст системи – взаємодію підприємства з зовнішнім середовищем;

– другий рівень описує основні напрями діяльності підприємства – тематично згруповані бізнес-процеси;

– подальша деталізація бізнес-процесів здійснюється за допомогою бізнес-функцій та основних бізнес-операцій, згрупованих за певними ознаками;

– опис елементарної господарської операції здійснюється шляхом визначення алгоритму її виконання.

Метод моделювання IDEF3 – частина сімейства стандартів IDEF; використовується для моделювання послідовності дій та їх взаємозалежностей у рамках процесу. Метод отримав визнання серед системних аналітиків як доповнення до методу функціонального моделювання IDEF0.

Модель IDEF3 заснована на сценарії процесу, який розділяє послідовність дій і підпроцесів системи. Як і в методі IDEF0, базовою одиницею моделі є діаграма. Іншим важливим компонентом є дія або «одиниця роботи» (Unit of Work), взаємодія яких зображується через зв'язки.

Діаграми потоків даних (Data Flow Diagrams – DFD) – це ієрархія функціональних процесів, пов'язаних потоками даних. Метою такого подання є демонстрація, як кожен процес перетворює свої вхідні дані у вихідні, а також виявлення зв'язків між цими процесами.

Відповідно до методу модель системи визначається як ієрархія діаграм потоків даних, основними компонентами якої є:

– зовнішні предмети;

– системи та підсистеми;

– процеси;

– пристрої зберігання даних;

– потоки даних.

Метод ARIS (Architecture of Integrated Information System) – це комплекс інструментів для аналізу та моделювання діяльності підприємства. Його методологічну основу складає сукупність різних методів моделювання, що відображають різні погляди на системи.

ARIS підтримує чотири типи моделей, які відображають різні аспекти досліджуваної системи:

– організаційні, що представляють структуру системи;

– функціональні, які містять ієрархію цілей;

– інформаційні – відображають структуру всієї інформації, необхідної для реалізації функцій системи;

– моделі управління, що представляють інтегрований підхід до реалізації бізнес-процесів усередині системи.

Для побудови цих типів моделей використовуються як власні методи моделювання ARIS, так і різні відомі методи та мови моделювання, зокрема UML.

Автори методу Ericsson-Penker створили свій UML-профіль для моделювання бізнес-процесів – Ericsson-Penker Business Extensions, вводячи набір стереотипів, що описують основні категорії бізнес-моделей: процеси, ресурси, правила та цілі підприємства.

UML також використовується в методі, який є частиною технології Rational Unified Process (фірми IBM). Цей метод спрямований, насамперед, на створення основи для формування вимог до програмного забезпечення. Передбачає побудову двох основних моделей:

– Модель бізнес-процесів (Business Use Case Model);

– Модель бізнес-аналізу (Business Analysis Model).

Business Use Case визначається як опис послідовності дій (потоків) у межах конкретного бізнес-процесу, який дає результат для конкретного суб'єкта.

3.5 Метод технології Rational Unified Process (UML)

Rational Unified Process (RUP) – це ітеративний процес розробки програмного забезпечення, створений Rational Software, підрозділом IBM з 2003 року. RUP – це не єдиний розпорядчий процес, а скоріше фреймворк процесу, яку організації з розробки та команди розробників адаптують для вибору елементів процесу, які відповідають їхнім потребам.

У 1997 році Rational придбала Verdix, Objectory, Requisite, SQA, Performance Awareness і Pure-Atria. Поєднання баз досвіду цих компаній привело до розробки семи «кращих практик» сучасної програмної інженерії:

1. Розробляти ітеративно, керуючись ризиками;
2. Управляти вимогами;
3. Використовувати компонентну архітектуру;
4. Моделювати програмне забезпечення візуально;
5. Постійно перевіряти якість;
6. Контролювати зміни;
7. Підлаштовуватись.

RUP пропонує ітеративний підхід до проєктування та розробки програмного забезпечення, заснований на спіральному життєвому циклі. Весь життєвий цикл має чотири фази – входження в проєкт (дослідження), розвиток (уточнення плану), конструювання та розгортання. Кожна фаза складається з послідовності ітерацій, число яких може бути будь-яким.

У кожній ітерації перераховані вище технологічні процеси послідовно застосовуються до розробки невеликої частини програмного забезпечення. При цьому допустимо показувати результат замовникові. Він має можливість оцінити виконану реалізацію, видати свої зауваження, які можуть привести до зміни та уточнення вимог до програмного забезпечення. Наступна ітерація передбачає розширення вже розробленої частини шляхом реалізації та інтеграції чергової порції вимог і врахування зміни вимог відповідно до зауважень замовника. Така організація процесу має цілу низку переваг:

– Зміни та уточнення вимог виявляються вже в ранній період розробки, коли обсяг програмного коду порівняно невеликий, тому трудомісткість внесення змін істотно нижча;

– Вже на ранній стадії процесу розробки є можливість залучення фахівців замовника для оцінювання проміжних версій (прототипів) програмного забезпечення. Як результат - значно більш висока ймовірність того, що кінцевий продукт буде робити саме те, чого чекає від нього замовник, тобто, гарантується висока якість програмного забезпечення;

– Знижуються архітектурні та інтеграційні ризики. При ітеративному підході створюється стійка архітектура, яка опрацьовується на стадіях дослідження, а потім перевіряється і уточняється в декількох початкових ітераціях. Кожна ітерація передбачає інтеграцію нових елементів в систему, тобто, кількість інтегрованих елементів в кожній ітерації невелика і легше простежується, ніж при глобальній інтеграції всіх розроблених елементів програмного забезпечення;

– Ітеративний підхід сприяє більш повному повторному використанню програмних елементів. Аналіз результатів кожної ітерації дозволяє архітекторам програмного забезпечення виділити фрагменти, які потенційно підлягають повторному використанню, а в наступній ітерації оформити їх як повторно використовувані коди.

RUP – процес, що направляється варіантами використання.

Поняття «use case», введене в мові UML, є основою для виконання всіх етапів життєвого циклу, що розглядаються в RUP. Поняття «business use case» (вид діяльності) є ключовим при бізнес-аналізі. На етапах аналізу вимог, проєктування і реалізації «use cases» виступають як варіанти використання системи, бувши тією «грубкою», від якої «танцюють» аналітики і розробники при виконанні проєктування і реалізації програмного забезпечення. При аналізі вимог, виділивши варіанти використання системи, ми тим самим визначаємо вимоги або рівень ієрархії вимог до програмного забезпечення. При деталізації варіантів використання системи визначаються об’єкти, і способи їх взаємодії, які мають бути реалізовані в програмному коді.

Потрібно особливо наголосити на ролі варіантів використання системи в плануванні ітерацій. Як визначити, яка функціональність має бути реалізована в черговий ітерації? Тут на виручку приходять діаграми варіантів використання системи. Кожному варіанту використання системи можна приписати пріоритет, який визначає, в якій ітерації його потрібно реалізувати. Можна показати всі варіанти використання системи, що реалізуються в черговий ітерації, на окремій діаграмі (або декількох діаграмах).

У RUP визначено 9 технологічних процесів, для кожного з яких запропонована методика виконання. Технологічні процеси діляться на дві категорії – основні процеси та процеси підтримки. До основних відносять:

– Бізнес-аналіз;
– Управління вимогами;
– Аналіз і проєктування;
– Реалізацію;
– Тестування;
– Розгортання.

Допоміжні процеси охоплюють:

– Управління проєктом;
– Управління конфігурацією;
– Управління середовищем.

Для кожного технологічного процесу передбачені ролі, що визначають поведінку і обов'язки окремих осіб і груп, які працюють в одній команді (наприклад, системний аналітик, тестувальник), види діяльності, що визначають роботи, які виконуються виконавцями (наприклад, проєктування класу, проєктування варіантів використання системи) і артефакти – документи, які використовуються, породжуються або модифікуються процесом. Основні артефакти в RUP – модель, елемент моделі, документ, вихідний код, що виконується програмою.

3.6 Метод Ericsson-Penker

Механізми UML призначені для того, щоб розробники могли адаптувати мову моделювання до своїх конкретних потреб, не змінюючи при цьому його модель. Мова UML принципово відрізняється від таких засобів моделювання, як IDEF0, IDEF1X, IDEF3, DFD тощо. Перераховані мови моделювання можна визначити як сильно типізовані (за аналогією з мовами програмування), оскільки вони не допускають довільної інтерпретації елементів моделей. UML, допускаючи таку інтерпретацію, є слабо типізованою мовою.

Метод Ericsson-Penker використовується при проєктуванні проєктів із застосуванням UML в рамках процесного підходу до моделювання бізнес-процесів. Автори методу створили свій UML-профіль для моделювання бізнес-процесів, ввівши набір стереотипів, що описують процеси, ресурси, правила та цілі організації. Метод використовує чотири основні категорії бізнес-моделей:

– Ресурси – різні об'єкти, що використовуються або задіяні в бізнес-процесах (люди, матеріали, інформація або продукти).

– Процеси – види діяльності, які змінюють стан ресурсів відповідно до бізнес-правил.

– Цілі – призначення бізнес-процесів можуть бути розбиті на підцілі і співвіднесені з окремими процесами.

– Бізнес-правила – умови або обмеження щодо здійснення процесів (функціональні, поведінкові або структурні). Правила можуть бути визначені з використанням мови об'єктних обмежень OCL (Object Constraint Language).

Основною діаграмою UML, що використовується в даному методі, є діаграма діяльності. Метод Eriksson-Penker є процесом у вигляді діяльності зі стереотипом «process» (основою даного подання є розширення методу IDEF0). Повна бізнес-модель містить безліч подань. Кожне подання виражено в одній або більше діаграмах UML. Діаграми можуть мати різні типи і зображати процеси, правила, цілі і ресурси у взаємодіях один з одним. Метод використовує чотири різних подання бізнес-моделі:

– концептуальне подання – структура цілей і проблем;

– подання процесів – взаємодія між процесами і ресурсами (у вигляді набору діаграм діяльності);

– структурне подання – структура організації і ресурсів (у вигляді діаграм класів);

– подання поведінки – поведінка окремих ресурсів і деталізація процесів (у вигляді діаграм діяльності, станів і взаємодії).

3.7 Основні принципи та розвиток моделювання бізнес-процесів

Моделювання бізнес-процесів у компанії може бути спрямоване на вирішення великої кількості різноманітних завдань:

– Точно визначити результат бізнес-процесу та його цінність для бізнесу;

– Визначити набір дій, з яких складається бізнес-процес. Чітке визначення набору завдань і дій, які необхідно виконати, надзвичайно важливо для детального розуміння процесу;

– Визначити порядок дій. Дії в рамках одного бізнес-процесу можуть виконуватися як послідовно, так і паралельно. Очевидно, що паралельне виконання, якщо воно допустиме, дозволяє скоротити загальний час виконання процесу і, як наслідок, підвищити його ефективність;

– Окремі сфери відповідальності: визначити, а потім відстежити, який співробітник або підрозділ компанії відповідає за виконання певної дії або процесу в цілому;

– Визначити ресурси, які споживає бізнес-процес. Коли точно знаєш, хто які ресурси і для яких операцій використовує, можна підвищити ефективність використання ресурсів за допомогою планування та оптимізації;

– Зрозуміти суть взаємодії між працівниками та підрозділами компанії, які беруть участь у процесі, та оцінити, а потім підвищити ефективність комунікації між ними;

– Бачити переміщення документів під час процесу. Бізнес-процеси виробляють і споживають різноманітні документи (у паперовій або електронній формі). Важливо розуміти, звідки і куди йдуть документи чи інформаційні потоки, а також визначити, чи є їх переміщення оптимальним і чи всі вони дійсно потрібні;

– Визначити потенційні вузькі місця та можливості для покращення процесу, які пізніше будуть використані для його оптимізації;

– Ефективніше впроваджувати стандарти якості та успішно проходити сертифікацію;

– Використовувати моделі бізнес-процесів як керівництво для нових співробітників;

– Ефективно автоматизувати бізнес-процеси в цілому або їх окремі кроки, зокрема автоматизацію взаємодії з зовнішнім середовищем – клієнтами, постачальниками, партнерами;

– Розуміння сукупності бізнес-процесів компанії, розуміння та опис діяльності підприємства в цілому.

Після визначення результату потрібно зрозуміти послідовність дій, з яких складається процес. Послідовність дій моделюється на різних рівнях абстракції. На верхньому рівні показують лише найважливіші етапи процесу (зазвичай не більше десяти). Потім виконується декомпозиція кожного з високорівневих кроків (підпроцесів).

На основі зібраної інформації будується модель звичайного або оптимального виконання процесу та визначаються можливі сценарії його виконання зі збоями. Різні збої (Exceptions – винятки) можуть порушити оптимальний хід процесу, тому потрібно вказати, яким чином будуть «оброблятися» винятки, тобто які дії виконуватимуться у разі винятку.

На рисунку 3.25 показані основні кроки при побудові моделі бізнес-процесу.

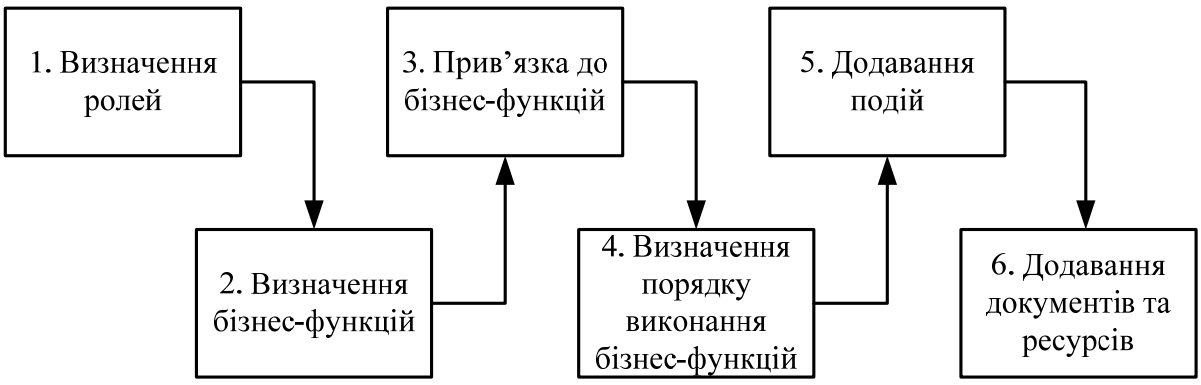


Рисунок 3.25 – Основні кроки при побудові моделі бізнес-процесу

Важливою частиною побудови моделі бізнес-процесу є дослідження аспектів його ефективності. Сюди входять: використання ресурсів, час виконання робіт співробітниками, можливі затримки і простої. Необхідно розробити систему показників або метрик для оцінювання ефективності процесу. Частково за метрики можуть бути взяті використовувані в компанії KPI (Key Performance Indicator), однак можуть знадобитися і додаткові показники, які характеризують даний процес.

Необхідно виявити події, які можуть перервати хід процесу. В разі переривання, можливо, потрібно буде коректно «відкотити» (компенсува-

ти) ті кроки процесу, які вже були виконані. Для цього потрібно визначити логіку компенсувальних дій для кожного переривання подій.

Нарешті, потрібно розглянути наявні програмні засоби, що здійснюють підтримку бізнес-процесу. Це важливо, тому що програмне забезпечення може приховувати деякі особливості поведінки процесу, не повністю відомі співробітникам, які виконують окремі кроки. Зібрана на цьому етапі інформація буде корисна при подальшій автоматизації процесу.

Зібравши всі зазначені відомості, можна отримати гарне уявлення про хід бізнес-процесу. На етапі моделювання мають бути отримані такі результати:

– Процесна карта, що показує зв'язок між різними бізнес-процесами і їх взаємодію. На процесній карті, як правило, кожен бізнес-процес компанії зображений у вигляді прямокутника, стрілками показані зв'язки між ними (наприклад, залежність одного процесу від іншого, або заміна одного процесу іншим, при виконанні деякої умови), а також подані різні документи, які передаються з процесу в процес або регламентують їх хід (стандарти, інструкції тощо);

– Діаграма ролей, що показує ролі при виконанні процесу і зв′язки між ними. Діаграма ролей не є ієрархічною. Вона показує такі зв'язки, як участь в групі, керівництво, комунікацію, заміщення однієї ролі іншого тощо;

Модель «як є» кожного розглянутого бізнес-процесу, детально описує процес і відображає хід процесу, дії, ролі, рух документів, а також точки можливої оптимізації. Така модель містить:

– Діаграму оточення процесу, що подає бізнес-процес у вигляді одної дії (тобто не розкриває хід процесу), для якого можуть бути показані: який процес запускає подія, необхідні вхідні дані, результат, ролі, показники ефективності, переривання події і компенсувальні процеси, регламентувальні документи, пов'язані бізнес-цілі;

– Високорівневу діаграму процесу, що показує його великі кроки (зазвичай не більше десяти) і пов'язані з ними ролі;

– Докладні діаграми для кожного кроку високорівневої моделі (залежно від складності процесу тут може використовуватися кілька ієрархічно організованих діаграм), які в деталях показують хід процесу, переривання події, бізнес-правила, ролі і документи;

– Діаграму обробки винятків, що показує, які дії виконуються в разі даної виняткової ситуації і ким, а також куди передається керування після закінчення обробки виняткової ситуації.

Контрольні питання

1) Які переваги функціонального моделювання бізнес-процесів (IDEF0)?

2) Опишіть загальні принципи методології IDEF0.

3) Яким чином можна графічно показати структури в IDEF0?

4) Назвіть приклади використання IDEF0.

5) Які переваги моделювання бізнес-процесів IDEF3?

6) Опишіть загальні принципи методології IDEF3.

7) Назвіть приклади використання IDEF0.

8) Наведіть приклади використання формалізованого опису бізнес-процесів.

9) В чому полягає суть моделювання потоків даних (DFD)?

10) Які особливості використання методу ARIS?

11) Які переваги та недоліки має метод технології Rational Unified Process?

12) Які переваги методу Ericsson-Penke?

13) Назвіть основні принципи моделювання бізнес-процесів.

14) Які перспективи розвитку моделювання бізнес-процесів?

ТЛУМАЧЕННЯ ОСНОВНИХ ТЕРМІНІВ

Аналіз (від грец. αναλυσις – «розклад») – розчленування предмета пізнання, абстрагування його окремих сторін чи аспектів. Метод дослідження, який вивчає предмет, уявно чи реально розчленовуючи його на такі складові елементи, як частини об'єкта, його ознаки, властивості, відношення, відтак розглядає кожен з виділених елементів окремо в межах єдиного цілого; протилежний метод – синтез.

Моделювання – це метод дослідження різних явищ і процесів, вироблення варіантів управлінських рішень. Моделювання ґрунтується на заміщенні реальних об'єктів їх умовними зразками, аналогами. Методом моделювання описуються: структура об'єкта (статична модель), процес його функціонування і розвитку (динамічна модель).

Бізнес-процес – будь-яка діяльність, що має вхідний продукт, додає вартість до нього та забезпечує вихідний продукт для внутрішнього або зовнішнього споживача. Існують три види бізнес-процесів:

– процеси управління – бізнес-процеси, які управляють функціонуванням системи. Прикладом керуючого процесу може служити корпоративне управління та стратегічний менеджмент.

– основні – бізнес-процеси, які складають основний бізнес компанії і створюють основний потік доходів. Прикладами операційних бізнес-процесів є постачання, виробництво, маркетинг та збут.

– забезпечувальні – бізнес-процеси, які обслуговують основний бізнес. Наприклад, бухгалтерський облік, кадрове, інформаційне забезпечення.

Бізнес-процес починається з попиту споживача і закінчується його задоволенням.

Бази даних (Databases) – це структурований набір даних про певні характеристики фізичних чи віртуальних систем.

Управління – це цілеспрямований програмований чи довільний вплив на об'єкти задля досягнення кінцевої мети за допомогою процесорів, явищ, процесів, коли є з ними взаємодія в режимі детермінованої чи довільної програми/регламенту. Управління проєктом/об'єктом-системою, її компонентами та процесами, з метою підвищення ефективності функціонування систем відбувається ще на етапі системного проєктування, створення/утворення, формування, розвитку, становлення, функціонування/життя системи. Ефективність управління визначається адекватністю дій управління щодо об'єкта управління. Управління є першим етапом тактичного рівня в алгоритмі системно-організаційній діяльності. Управління є координувальним, адміністративним, виконавчим рівнем в тактиці. Стратегічне управління пов'язане з аналізом проблем, виявленням крайових умов задач, пошуку оптимальних моделей рішень, забезпечення виконавчими процесорами, ресурсами, програмою дій, зокрема керівними проце-

сорами чи кадровими управлінцями для тактичного, виконавчого рівня проєкту/об'єкта-системи.

ІТ-проєкт – це процес створення продукту. Це те, що робить команда, щоб видати замовнику продукт. Продукт – те, що хоче отримати замовник. Продукт – результат (чи набір результатів) поставки по контракту. Метою будь-якого проєкту є отримання кінцевого продукту.

Інформаційна система – сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів.

Інформаційна технологія – сукупність методів, виробничих процесів і програмно-технічних засобів, інтегрованих з метою збирання, опрацювання, зберігання, розповсюдження, показу і використання інформації в інтересах її користувачів. Технології, що забезпечують та підтримують інформаційні процеси, тобто процеси пошуку, збирання, передачі, збереження, накопичення, тиражування інформації та процедури доступу до неї.

Система (System) – це сукупність елементів, певним чином пов'язаних і таких, що взаємодіють між собою, для виконання заданих цільових функцій.

Цифрова карта (Digital map) – цифрова модель місцевості, створена шляхом оцифровування картографічних джерел, фотограмметричної обробки даних дистанційного зондування, цифрової реєстрації даних польових зйомок або іншим способом.

Електронна карта (Electronic map) − картографічне зображення, яке візуалізоване на дисплеї (екрані) комп'ютера на основі даних цифрових карт або баз даних ГІС з використанням програмних і технічних засобів у прийнятій для карт проєкції і системі умовних знаків.

ЛІТЕРАТУРА

1. Моделювання бізнес-процесів та управління ІТ-проєктами : електронний навч. пос. / Крижановський Є. М., Ящолт А. Р., Жуков С. О., Козачко О. М. Вінниця : ВНТУ, 2018. 91 с.
2. Моделі управління проєктами: рекомендації до виконання розрахункової роботи [Електронний ресурс] : навч. посіб. для студ. денної та заочної форм навчання другого магістерського рівня вищої освіти спеціальності 051 «Економіка» освітньо-професійної програми «Економічна кібернетика» / КПІ ім. Ігоря Сікорського ; уклад.: О. П. Кавтиш. Електронні текстові данні (1 файл: 755 Кб). Київ : КПІ ім. Ігоря Сікорського, 2020. 60 с.
3. Методичні вказівки до практичних занять з дисципліни «Методологія та методи управління проєктами в галузі ІТ» для студентів усіх форм навчання другого (магістерського) рівня вищої освіти спеціальності 122 «Комп'ютерні науки» освітньо-професійної програми «Управління проєктами в галузі інформаційних технологій» [Електронне видання] / Упоряд. М. В. Євланов. Харків : ХНУРЕ, 2019.
4. Scrum. Навчись робити вдвічі більше за менший час / Джефф Сазерленд. Business must read – КСД, 2022. 280 с.
5. Методичні вказівки до самостійної роботи студентів з дисципліни «Методологія та методи управління проєктами в галузі ІТ» для студентів усіх форм навчання другого (магістерського) рівня вищої освіти спеціальності 122 «Комп'ютерні науки», освітньо-професійної програми «Управління проектами в галузі інформаційних технологій» [Електронне видання] / Упоряд. М. В. Євланов. Харків : ХНУРЕ, 2019.
6. Дружинін Є.А. Методологічні основи ризик-орієнтованого підходу до управління ресурсами проектів і програм розвитку техніки : автореф. дис. на здобуття наук. ступеня д- ра техн. наук : 05.13.22. Х., 2006. 34 с.
7. Катренко А. В., Рішняк І. В. Методи управління ризиками в ІТ-проєктах. *Комп'ютерні науки та інформаційні технології (CSIT-2008):* III Міжнар. наук.-практ. конф., 25–27 вересня 200 8р.: тези доповіді. Львів, 2008. С. 245–247.
8. ВересО. М., Катренко А. В., Рішняк І. В., Чаплига В. М. Управління ризиками в проєктній діяльності. *Інформаційні системи та мережі, Вісн. Нац. ун-ту «Львівська політехніка»*. 2003. № 489. С. 38–49.
9. Плескач В. Л., Затонацька Т. Г. Інформаційні системи і технології на підприємствах : підручник. К. : Знання, 2011. 718 с.
10. Крижановський Є. М., Мокін В. Б., Ящол А. Р., Скорина Л. М. Вінниця : Системний аналіз та проєктування ГІС :електронний навч. пос. ВНТУ, 2015. – 127 с. – Режим доступу: http://ir.lib.vntu.edu.ua/bitstream/handle/123456789/8960/Posibnik_2015_3%20(1).pdf?sequence=1

11. Катренко А. В., Рішняк І. В. Методи управління ризиками в ІТ-проєктах. *Комп'ютерні науки та інформаційні технології (CSIT-2008):* III Міжнар. наук.- практ. конф., 25–27 вересня 200 8р. : тези доповіді Львів, 2008. С. 245–247.

12. Рішняк І. В. Модель управління проєктними ризиками. Комп'ютерні системи проєктування. Теорія і практика. *Вісник Національного університету «Львівська політехніка»*.2004. № 522. С.155–160.

13. Designing the Business Models for Circular Economy – Towards the Conceptual Framework / M. Lewandowski // Sustainability, vol. 8(1):43, 2016. DOI:10.3390/su8010043.

14. Верес О. М., Катрено А. В., Рішняк І. В., Чаплига В. М. Управління ризиками в проєктній діяльності. *Інформаційні системи та мережі* : // Вісн. Нац. ун-ту «Львівська політехніка». 2003. – № 489. С. 38–49.

15. Project management institute. [Електронний ресурс] // Режим доступу : http://www.pmi.org/.

16. Стандарти управління проєктами. [Електронний ресурс] // Режим доступу:http://studme.org/1055120821033/menedzhment/standarty_upravleniya_proektami

17. Каскадна модель. [Електронний ресурс] // Режим доступу : http://asset.in.ua/novosti-ukrainy/item/13713-1453915397.

18. Документация проєкта. [Электронный ресурс] // Режим доступу : :http://www.pmphelp.net/index.php?id=33.

Електронне навчальне видання
комбінованого використання.
Можна використовувати в локальному та мережному режимах

Крижановський Євгеній Миколайович
Яцолт Андрій Русланович
Жуков Сергій Олександрович

МОДЕЛЮВАННЯ БІЗНЕС-ПРОЦЕСІВ ТА УПРАВЛІННЯ IT-ПРОЄКТАМИ

Навчальний посібник

Рукопис оформлено *С. Жуковим*

Редактор *В. Дружиніна*

Оригінал-макет виготовлено *О. Ткачуком*

Підписано до видання 05.10.2022
Гарнітура Times New Roman.
Зам. № Р2022-080.

Видавець та виготовлювач
Вінницький національний технічний університет,
редакційно-видавничий відділ
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95, м. Вінниця, 21021.
Тел. (0432) 65-18-06.
press.vntu.edu.ua;
E-mail: kivc.vntu@gmail.com
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.