МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНИЙ ПОСІБНИК

КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ ОБРОБКИ МУЛЬТИМЕДІЙНОЇ ІНФОРМАЦІЇ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

Чичур А.І., Кравчук П.О., Волуйко І.В.

Затверджено Вченою радою навчально-наукового інституту Інформаційних технологій ДУІКТ як навчальний посібник для студентів вищих навчальних закладів

Київ – 2025

УДК 004.9 (075.8)

Гриф надано Вченою радою навчально-наукового інституту Інформаційних технологій ДУІКТ
(протокол № 04 від 11.06.2025 р.)

Рецензенти: д.т.н., проф. Мухін В.Є., к.т.н., доц. Пампуха І.В.

Чичур А.І., Кравчук П.О., Волуйко І.В.

Комп’ютерні технології обробки мультимедійної інформації на основі ШІ. Навчальний посібник підготовлено для самостійної роботи студентів та аспірантів вищих навчальних закладів. Київ: ННІІТ ДУІКТ, 2025. – 138 с.

Навчальний посібник «Комп’ютерні технології обробки мультимедійної інформації на основі ШІ» має за мету надати студентам теоретичні знання і практичні навички з основ інформаційних систем, історії їх розвитку та класифікації.

Управлінні проєктами з розробки та впровадження інформаційних систем та програмних продуктів із використанням сучасних методологій, фреймворків і штучного інтелекту.

Техніки проведення бізнес-аналізу для визначення потреб зацікавлених осіб проєкту, методи та моделі, що описують бізнес-процеси, засвоїти принципи документування змін при розробці програмних продуктів.

Основи концепції дизайну програмних продуктів та методів його розробки, а також засвоїти теоретичні основи використання технологій штучного інтелекту (англ. artificial intelligence, AI) та бізнес-аналізу (business intelligence, BI) для обґрунтованого прийняття бізнес-рішень та зменшення рутинних задач.

Основні поняття веб-розробки, мінімальний життєздатній продукт (MVP), застосування програмного забезпечення із відкритим вихідним кодом, різні види ліцензування програмних продуктів.

Створення інформаційних систем за допомогою інструментів «без коду» або «з низьким кодом» (англ. Low-Code / No-Code, LCNC), в поєднанні із ШІ.

Управління змінами. Управління ризиками. Ретроспектива. Управління командами. Управління ресурсами.

Навчальний посібник призначений для студентів, які навчаються за технічними спеціальностями в галузях 12 Інформаційні технології, 17 Електроніка та радіотехніка, 07 Управління та адміністрування а також може бути корисний для аспірантів, викладачів навчальних закладів відповідних спеціальностей.

ЗМІСТ

ПЕРЕДМОВА

Сучасний світ неможливий без інформаційних систем та технологій. В умовах цифрової економіки компанії використовують величезні обсяги даних для аналізу ринку, прогнозування поведінки споживачів і автоматизації бізнес-процесів.

Інформаційні системи та технології широко використовуються в промисловості, господарстві, науці і техніці, сфері послуг, маркетингу, тощо. Вони дозволяють підвищити ефективність та оптимізувати рутинні процеси, контролювати всі стадії виробництва та надання послуг, покращити взаємодію з клієнтами, впливати на рівень продажів, швидко опрацьовувати великі обсяги даних, управляти промисловим устаткуванням, обробляти результати досліджень, тощо.

Розробка інформаційних систем важкий і енергоємний процес, що вимагає кваліфікованої команди із достатнім рівнем компетенцій. Важливу увагу приділяють організації процесу розробки програмних продуктів та інформаційних систем.

1. ОСНОВИ ІНФОРМАЙНИХ СИСТЕМ

1.1 Загальні відомості про інформаційні системи

Інформаці́йна систе́ма (англ. Information system) – це комплекс технічних та організаційних засобів спрямованих на збереження і обробку інформації для забезпечення потреб користувачів. Також визначають її, як систему, що забезпечує збирання, пошук, оброблення та пересилання інформації.

Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» визначає інформаційну (автоматизовану) систему як організаційно-технічну систему, в якій реалізується технологія обробки інформації з використанням технічних і програмних засобів.

Загалом, така система, призначена для одержання, обробки, зберігання, відображення та/або реєстрації даних про технічний стан конструкцій, систем, елементів, їх властивості та/або функціонування.

Спочатку інформацією вважалися будь-які відомості, що передаються людьми усно, письмово або в інший спосіб (сигнали, засоби тощо). Проте, зараз вважається, що інформаційна взаємодія є в основі процесів керування у системах будь-якої природи (походження). При цьому розрізняють *синтаксичний*, *семантичний* й *прагматичний* аспекти інформації.

Синтаксичний характеризує план її виразу, склад, структуру, складність, організованість. Семантичний визначає її зміст та значення (інформація – це відображення чого-небудь, тобто вона цікава не сама по собі, а як представник об'єкта чи явища). Прагматичний характеризує її здатність впливати на процеси управління у системі з точки зору її цінності, користі або шкоди (з точки зору інтересів того, хто займається творенням й дистрибуцією інформації).

Інформаційні системи вже дуже давно представлені різноманітними формами в життєдіяльності людини. Широке застосування пов’язано з необхідністю обмінюватись інформацією, як запорукою існування цивілізації. Простіше кажучи, потреба в реалізації задачі передачі знань: між членами сім’ї, соціальними група, общинами, поколіннями, тощо.

Інформаційні системи існують з моменту появи суспільства, оскільки на кожній стадії його розвитку існує потреба в управлінні. Місією інформаційної системи є переробка інформації, потрібної для ефективного управління всіма ресурсами організації, створення інформаційного та технічного середовища для управління її діяльністю.

Така система, звісно, може існувати і без застосування комп'ютерної техніки, але це питання економічної доцільності. В наш час як раз розвиток технологій сприяє ефективному впровадженню інформаційних систем.

В будь-якій інформаційній системі управління вирішуються три типи задач:

- оцінки ситуації (або розпізнавання образів);
- перетворення опису ситуації (розрахункові задачі, задачі моделювання);

- прийняття рішень (в тому числі і оптимізаційні).

Інформаційні системи класифікують за ознакою структурованості задач. При розробці ІС виникає проблема з формальним математичним і алгоритмічним описом розв'язуваних задач. Ступень формалізації (рівень організованості) визначає ефективність роботи всієї системи, а також рівень автоматизації, обумовлений ступенем участі людини при ухваленні рішення на основі одержуваної інформації.

Отже, чим точніший математичний опис задачі, тим вищі можливості комп'ютерної обробки даних і тим менше ступінь участі людини в процесі її рішення, що визначає ступінь автоматизації задачі.

Відповідно, розрізняють три типи задач для вирішення яких створюють інформаційні системи:

- *структурована* – зміст виражено у формі математичної моделі, що має алгоритм рішення. Такі задачі вирішуються багаторазово, оскільки мають рутинний характер. Метою використання інформаційної системи для рішення структурованих задач є повна автоматизація їхнього рішення, тобто зведення ролі людини до нуля.;
- *неструктурована* – зміст неможливо виразити у формі математичної моделі, а розробка алгоритму зв'язано з великими труднощами. Такі задачі мають обмежені можливості використання, а вирішення приймається людиною з евристичних розумінь на основі свого досвіду і, можливо, непрямої інформації з різних джерел;
- *частково (або погано) структурована* – більшість задач мають лише частково виражений зміст і відомо лише частину їхніх елементів і зв'язків між ними. Інформаційні системи, створені для вирішення таких задач, є автоматизованими, оскільки в їхньому функціонуванні бере участь людина.

Розглянемо кілка прикладів:

Приклад 1. Необхідно реалізувати задачу розрахунку заробітної плати для працівників підприємства в інформаційній системі. Така задача є *структурованою*, оскільки цілком відомо алгоритм її рішення. Він базується на основі нормативно-правових актів та інструкцій. Така задача є рутинною, бо розрахунки всіх нарахувань і відрахувань дуже прості, але їх обсяг великий, тому що вони повинні багаторазово повторюватися щомісяця для всіх категорій працюючих.

Приклад 2. Необхідно формалізувати (організувати, описати) всі взаємини в студентській групі або колективі. Скоріш за все вирішення такої задачі неможливо тому, що психологічний і соціальний фактори дуже складно описати алгоритмічно, що робить її *неструктурованою* задачею.

Приклад 3. Необхідно вирішити задачу із усунення ситуації коли не вистачає трудових ресурсів для виконання роботи і шляхами для вирішення може бути: виділення додаткового фінансування; перенесення терміну закінчення роботи.

Як видно, у даній ситуації інформаційна система може допомогти людині прийняти те чи інше рішення, якщо надасть йому інформацію про хід виконання робіт із усіх необхідних параметрів, що робить її *частково структурованою* задачею.

В свою чергу, інформаційні системи, які використовуються для вирішення частково структурованих задач, підрозділяються на два види:

- такі, що створюють управлінські звіти, орієнтовані головним чином на обробку даних (пошук, сортування, агрегування, фільтрацію). Спираючись на дані в цих звітах, керуючий приймає рішення;
- можливі альтернативи рішення, що розробляють. Прийняте рішення при цьому зводиться до вибору однієї з запропонованих альтернатив.

Інформаційні системи, *що розробляють альтернативи рішень,* можуть бути *модельними* (надають користувачу математичні, статистичні, фінансові та інші моделі, які полегшують розробку й оцінку альтернатив рішення) чи *експертними* (забезпечують розробку й оцінку альтернатив рішення користувачем за рахунок створення експертних систем, зв'язаних з обробкою знань).

Модельні інформаційні системи характеризуються швидкою та адекватною інтерпретацією результатів моделювання, оперативною підготовкою та коригуванням вхідних параметрів.

Експертна підтримка прийнятих користувачем рішень реалізується на двох рівнях. Перший рівень базується на концепції «типових управлінських рішень», тобто до деякого типового набору альтернатив (створення інформаційного фонду). Якщо ж застосування типових альтернатив не можливе, то варто застосовувати другий рівень експертної підтримки управлінських рішень. Цей рівень генерує альтернативи на базі наявних в інформаційному фонді даних, правил перетворення і процедур оцінки синтезованих альтернатив.

Інформаційні системи також класифікують за функціональною ознакою, яка визначається видом діяльності. Типовими видами діяльності є: виробнича, маркетингова, фінансова та кадрова.

Виробнича діяльність пов'язана з виготовленням продукції, створенні науково-технічних нововведень.

Маркетингова діяльність пов'язана з аналізом ринку (виробники, споживачі, товари та послуги, аналіз продажів) та організацію рекламної кампанії.

Фінансова діяльність пов'язана з організацією контролю фінансових ресурсів підприємства на основі бухгалтерської, статистичної, оперативної інформації.

Кадрова діяльність спрямована на контроль людських ресурсів та організацією службової документації.

Тип інформаційної системи залежить від того, чиї інтереси і на якому рівні керування вона обслуговує. Класифікація інформаційних систем за

функціональною ознакою з врахуванням рівнів керування та кваліфікації персоналу зображено на Рис. 1.1.

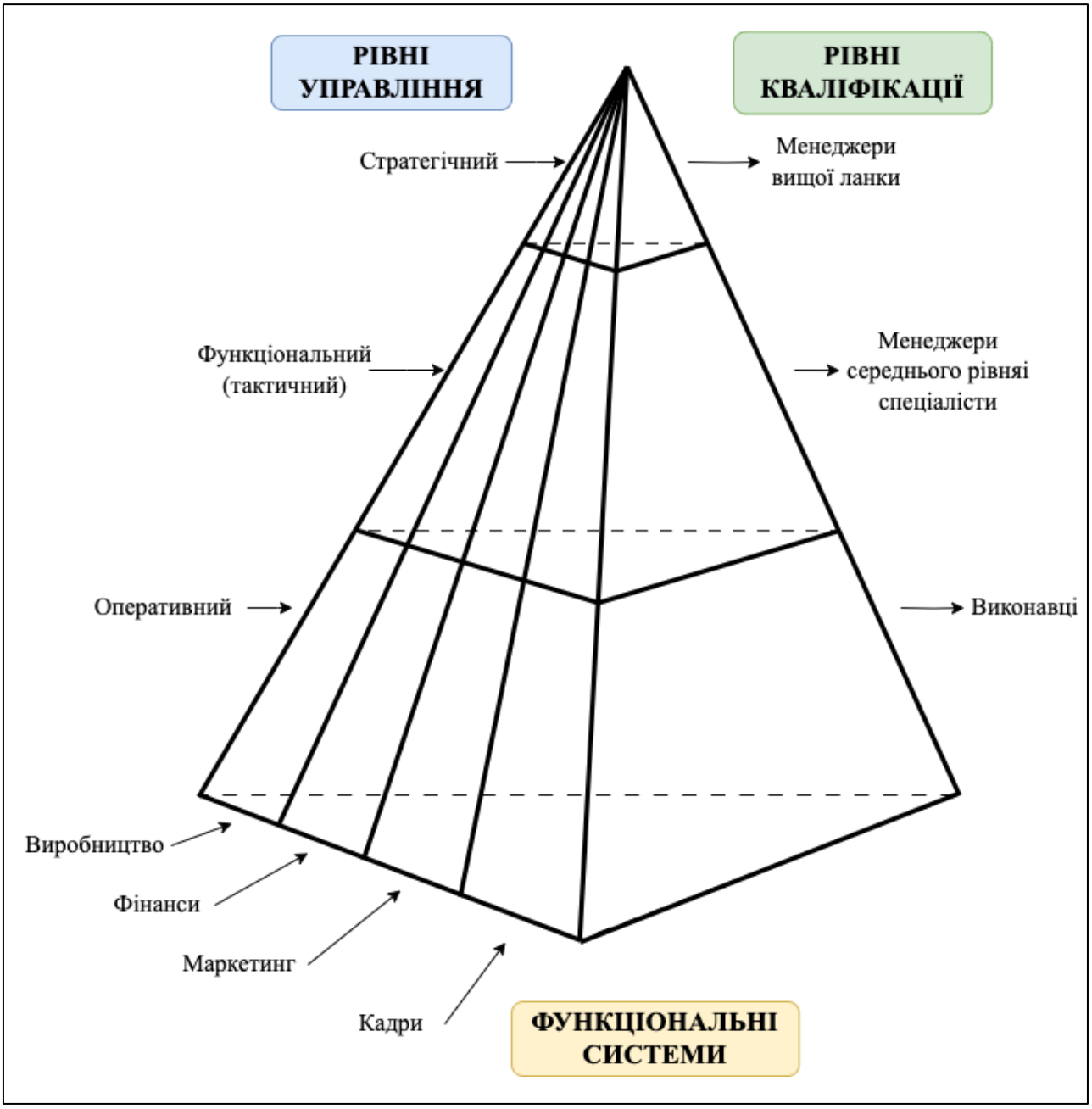

Рисунок 1.1 – Типи інформаційних систем залежно від функціональної ознаки

Основу піраміди складають інформаційні системи, за допомогою яких співробітники-виконавці займаються операційною обробкою даних, а менеджери нижчої ланки оперативним керуванням. Нагорі піраміди на рівні стратегічного керування інформаційні системи змінюють свою роль і стають стратегічними, підтримуючи діяльність менеджерів вищої ланки по прийняттю рішень в умовах поганої структурованості поставлених задач.

Кожен рівень управління має свою потребу в інформації але в різних обсягах та з певним рівнем абстракції чи узагальнення. Як видно з рисунка 1.1, чим

вищий рівень управління, тим менше обсяг роботи для виконавця, проте підвищується складність. Також, вплив прийнятих рішень на вершині цієї піраміди має значні наслідки.

В сучасному світі інформаційні системи певного функціонального призначення може складатися з кількох структурних елементів – підсистем. Також, часто застосовують терміни «модуль», «плагін», «додаток», тощо. Наприклад, виробнича інформаційна система має такі підсистеми: керування запасами, керування процесами виробництва, тощо.

Для кращого розуміння функціонального призначення інформаційних систем у табл.1.1 наведені по кожному розглянутому вище типу розв'язувані в них типові задачі.

Таблиця 1.1 – Типові задачі відповідних ІС

Тип ІС	**Типові задачі**
Виробничі	▪ Планування робіт (календарні плани, специфікації, технологічні карти); ▪ Оперативний контроль над процесом виробництва; ▪ Аналіз роботи виробничого обладнання; ▪ Формування замовлень постачальникам; ▪ Керування запасами.
Фінансові	▪ Керування портфелем замовлень; ▪ Керування кредитною політикою; ▪ Розробка фінансового плану; ▪ Фінансовий аналіз і прогнозування; ▪ Контроль бюджету; ▪ Бухгалтерський облік і розрахунок зарплати.
Маркетингові	▪ Дослідження ринку й прогнозування продажів; ▪ Керування продажами; ▪ Рекомендації з виробництва нової продукції; ▪ Аналіз і встановлення ціни; ▪ Облік замовлень.
Кадрові	▪ Аналіз і прогнозування потреби в трудових ресурсах; ▪ Ведення архівів записів про персонал; ▪ Аналіз і планування підготовки кадрів.
Інші (наприклад ІС керівництва)	▪ Контроль за діяльністю фірм, групи компаній; ▪ Виявлення оперативних проблем; ▪ Аналіз управлін-ських і стратегічних ситуацій; ▪ Забезпе-чення процесу вироблення стратегічних рішень.

1.2 Історія розвитку інформаційних систем

Найдавнішими і найпоширенішими ІС є бібліотеки, оскільки, здавна в них накопичували і зберігали книги, манускрипти, рукописи із дотриманням певної системи їх обліку, створення каталогу для полегшення пошуку в книжковому фонді.

Найстарші інформаційні системи повністю базувалися на ручній праці. Пізніше їм на зміну прийшли різні механічні пристрої для обробки даних (наприклад, для сортування, копіювання, асоціативного пошуку тощо).

Наступним кроком стало впровадження автоматизованих інформаційних систем (АІС), тобто систем, де для забезпечення інформаційних потреб користувачів використовується електронно-обчислювальна машина (ЕОМ) зі своїми носіями інформації.

Сьогодні, в епоху інформаційної революції – розробляється і впроваджується велика кількість найрізноманітніших АІС-ів з дуже широким спектром використання.

Інформаційні системи почали відігравати значну роль у повсякденні ще у другій половині XX століття, коли компанії почали використовувати комп'ютери для аналізу даних про клієнтів, планувати бюджети, виробничі процеси, контролювати якість, та накопичувати й управляти інформацією (даними), створюючи бази даних.

У 1950-х роках з'явилися перші інформаційні системи, що представляли собою електромеханічні бухгалтерські рахункові машин. Використовувались для обробки рахунків та розрахунку заробітної плати. Така автоматизація суттєво зменшувала витрати і час на підготовку паперових документів та дещо мінімізували помилки.

У 1960-і роки інформаційні системи значно збільшили сферу свого застосовування. Їх використовують у фінансовій звітності із багатьма параметрами, що стало поштовхом для розробки комп'ютерного устаткування із широким спектром призначення. Найвідомішим устаткуванням став ряд програмно-сумісних ЕОМ IBM/360, який створила фірма International Business Machines (IBM).

Тоді ж IBM, почала розробляти перші автоматизовані системи управління маркетинговими кампаніями, що базувалися на обробці великих масивів інформації. А одним із перших дослідників у цій сфері був американський вчений-маркетолог, професор міжнародного маркетингу Вищої школи менеджменту імені Дж. Л.Келлога при Північно-Західному університеті в США Філіп Котлер, який розробив теоретичні основи маркетингового аналізу.

У 1970-х – на початку 1980-х років інформаційні системи почали використовувати як інструмент управлінського контролю. А згодом вони стають джерелом інформації, необхідної на всіх рівнях організації, надаючи потрібну

інформацію користувачеві, що допомагає знаходити нові ринки збуту, створювати нові товари та послуги, тощо.

У 1980-х роках розвиток персональних комп'ютерів та баз даних дозволив маркетологам більш ефективно аналізувати споживчу поведінку. Компанія Oracle, заснована Ларрі Еллісоном, створила потужні системи управління базами даних, які широко використовувалися для маркетингових досліджень. Крім того, у цей період почався розвиток програмного забезпечення для бізнес-аналітики, наприклад, SAS (Statistical Analysis System) та SPSS (Statistical Package for Social Science).

В 1990-х роках починається ера переосмислення застосування інформаційних систем в бізнесі та побуті. Вектор розвитку спрямований на глобалізацію суспільства. Розвиток цифрових та телекомунікаційних систем призводить до об’єднання обчислювальних ресурсів людства в єдину мережу – Інтернет.

З його появою дуже швидко зростає його роль в житті людини і всього суспільства разом, а згодом відбувся стрімкий розвиток цифрового маркетингу, що дозволив компаніям взаємодіяти з клієнтами у режимі реального часу. У 1994 році Джефф Безос заснував Amazon, який став одним із перших онлайн-магазинів, що активно використовував аналітичні системи для персоналізації пропозицій клієнтам. Водночас з’явилися перші онлайн-рекламні платформи, такі як DoubleClick, заснована Кевіном О'Коннором, яка використовувала нові методи таргетованої реклами.

У 2000-х роках соціальні мережі, такі як Facebook (заснований Марком Цукербергом у 2004 році), стали новим каналом маркетингових комунікацій. Водночас компанія Google запустила платформу Google AdWords, що дозволила підприємствам ефективно використовувати контекстну рекламу. У цей період також активно розвивалися CRM-системи (Customer Relationship Management - система управління відносинами з клієнтами), такі як Salesforce (заснований Марком Беніоффом), які стали ключовими для управління взаємодією з клієнтами.

З 2010-х років масове впровадження Big Data, штучного інтелекту та машинного навчання значно змінило маркетингові стратегії. Компанії, такі як Adobe, запустили потужні маркетингові платформи, а Netflix та Amazon почали використовувати алгоритми машинного навчання для персоналізації контенту. Паралельно Google розвинула технологію прогнозної аналітики, що дало змогу маркетологам краще прогнозувати поведінку споживачів.

Сьогодні інформаційні системи охоплюють широкий спектр технологій: від фінансових, облікових і аналітичних платформ до автоматизованих CRM-систем, які допомагають збирати, обробляти та використовувати дані про клієнтів. А також мобільних та WEB-застосунків, що стали невід’ємною частиною нашого побуту.

2. УПРАВЛІННЯ РОЗРОБКОЮ ТА ВПРОВАДЖЕННЯМ ІНФОРМАЦІЙНИХ СИСТЕМ

2.1 Системний підхід в розробці інформаційних систем

Розробка та впровадження інформаційних систем є дуже важливими для успіху бізнесу. Правильно розроблені інформаційні системи можуть зробити організацію більш гнучкою та ефективною, а недоліки в них навпаки, можуть мати шкідливий вплив на бізнес-ефективність (тобто зниження доходів, збільшення зобов'язань, втрати продуктивності та пошкодження репутації бренду) і навіть можуть призвести до краху бізнесу.

Розробка інформаційної системи необхідна, коли:

- потенційна конкуренція та розвиток технологій призводять до розробки нової системи для виконання вже існуючої функції;
- існуючі системи модифікуються для виконання додаткових функцій системи.

Система повинна пройти повний свій життєвий цикл (тобто від фази «Розробка» до фази «зниження корисності»). Саме така циклічність визначає передумови необхідності розробки інформаційної системи.

Системний підхід – метод дослідження (вирішення проблем), що полягає в фокусуванні на об'єкті як на цілісній системі, що складається з взаємопов'язаних елементів, а не як сукупність окремих частин. Він передбачає вивчення об'єкта у контексті його середовища та взаємодії з іншими системами, враховуючи внутрішні та зовнішні зв'язки. Системний підхід – це «бачити і дерева і ліс» (системна орієнтація).

Саме такий підхід лежить в основі філософії розробки чи впровадження інформаційних систем. Системному підходу характерні такі властивості характеристики:

- Бачення взаємозв'язків між системами, а не лінійних причинно-наслідкових ланцюгів, коли відбуваються події;
- Бачення процесу змін між системами, а не дискретних знімків змін щоразу, коли відбуваються зміни;
- Застосування системного контексту: визначення систем, підсистем та компонентів систем.

Існує п'ять основних етапів системного підходу:

1. Визначення проблеми/можливості за допомогою системного мислення;
2. Розробка та оцінка альтернативних системних рішень;
3. Вибір системного рішення, яке найкраще відповідає вашим вимогам;
4. Проектування вибраних системних рішень відповідно до ваших вимог;
5. Впровадження та оцінка успіху розробленої системи.

2.2 Життєвий цикл розробки інформаційних систем

Процес розробки інформаційних систем дуже важливий, як з економічної так і з технічної точки зору, оскільки має на меті створення високоякісного програмного забезпечення. Цей процес схожий для всіх інформаційних систем, тому його називають життєвим циклом.

Життєвий цикл розробки інформаційних систем (англ. Systems Development Life Cycle, SDLC) – багатоетапний процес розробки інформаційних систем для вирішення бізнес-задач та бізнес-проблем, що оснований на системному підході. Він включає етапи планування, аналізу, проектування, розробки, тестування, впровадження та підтримку системи.

Життєвий цикл розробки інформаційних систем є важливою складовою успішної розробки програмного забезпечення, оскільки допомагає забезпечити ефективність, якість та керованість процесу розробки, зменшуючи ризики та забезпечуючи досягнення поставлених цілей.

Організованість процесу розробки ІС може описуватись різними моделями або методологіями. У цих методологіях (моделях) описується кілька етапів, які поділяють процес розробки на завдання, які можна розподіляти, виконувати і оцінювати.

Модель життєвого циклу розробки ІС концептуально представляє SDLC в організованому вигляді, щоб допомогти впровадити ІС. Різні моделі мають фази SDLC в різному хронологічному порядку для оптимізації циклу розробки. Нижче ми розглянемо деякі популярні моделі SDLC.

Каскадна модель – всі етапи розташовані послідовно, тому кожен новий етап залежить від результатів попереднього. Розробка ІС відбувається шляхом переходу від однієї фази до іншої, подібно до каскаду (або водоспаду). Така модель суттєво дисциплінує процес розробки, проте після завершення етапу залишається мало можливостей змінити щось, оскільки ці зміни суттєво вплинуть на терміни наступних етапів та постачання в цілому. Тому така модель найбільше підходить для невеликих об’ємів робіт, із заздалегідь чітко визначеними вимогами.

Ітеративна модель – характеризується невеликою кількістю вимог до майбутньої ІС, що обумовлено філософією покращувати програмне забезпечення із кожною ітерацією, до поки не буде готове для використання. Тобто, кожною ітерацією створюється нова версія ПЗ (схоже на те як ліпити сніговика). Така модель дозволяє легко виявляти ризики та керувати ними, оскільки вимоги можуть змінюватися між ітераціями. Однак, повторювані цикли, можуть суттєво збільшити обсяг робіт.

Спіральна модель – поєднує в собі невеликі цикли ітеративної моделі, з лінійним і послідовним потоком каскадної моделі для визначення пріоритетності аналізу ризиків. Такий «гібрид», що поєднує характеристики каскадної та

ітеративної моделі, підходить для великих та складних задач, що потребують частих змін. І водночас, така модель може бути дорогою для невеликих об’ємів робіт із обмеженим масштабом.

Гнучка модель – характеризується розподілом етапів SDLC на кілька циклів розробки, що дозволяє швидко пройти усі етапи ітерацій, вносячи у кожному циклі лише невеликі додаткові зміни в ПЗ. Також, відбувається постійна оцінка вимог, планів та результатів, що дозволяє швидко реагувати на зміни, а отже бути гнучким. Така модель є ітеративною та поступовою одночасно, а значить більш ефективною в порівнянні з іншими моделями процесів.

Цикли швидкої розробки допомагають виявляти та вирішувати проблеми на ранніх стадіях і до того, як вони стануть серйозними. Однак надмірна залежність від відгуків та оцінок щодо вимог та планів може призвести до непотрібного збільшення обсягу робіт.

На основі моделей SDLC створюють методології розробки та впровадження інформаційних систем.

Екстремальне програмування (Extreme Programming, XP) – це гнучка методологія розробки ПЗ, яка характеризується швидкістю, ітеративністю, адаптивністюь до змін та орієнтована на покращення якості ПЗ. Їй притаманні такі властивості (принципи):

- парне програмування: два виконавця працюють разом над однією задачею, що сприяє обміну знань;
- розробка через тестування (Test-Driven Development, TDD): тести створюються перш ніж функціональність;
- часті невеликі випуски: ПЗ випускається частинами у коротких циклах, що дозволяє швидко отримувати зворотній зв'язок;
- рефакторинг: ПЗ постійно покращується для підтримки його якості та зручності;
- слухання (Listening): регулярний збір зворотного зв'язку для забезпечення відповідності ПЗ його потребам.

Екстремальне програмування, як гнучка методологія, зосереджується на людях та їх взаємодії, а не на процесах та інструментах, що робить її ефективною з невизначеними або швидкозмінними вимогами.

Методологія швидкої розробки додатків (Rapid Application Development, RAD) – це гнучка методологія розробки ПЗ, що акцентує увагу на швидкому створенні прототипів та його ітеративному вдосконаленні. Вона націлена на скорочення часу розробки та спрощення процесу впровадження змін, не жертвуючи якістю. Їй притаманні такі властивості (принципи):

- швидкість та ефективність: передбачає використання готових компонентів, інструментів та технік для прискорення процесу розробки;

- ітеративність: розробка відбувається ітераціями, зміни вносяться на основі зворотного зв'язку;
- прототипування: прототипи створюються на ранніх етапах розробки ПЗ, що дозволяє швидко перевіряти ідеї та вносити зміни.

Такий підхід суттєво скорочує терміні розробки ПЗ, шляхом зменшення витрат, проте, не нехтуючи якістю та продуктивностю. Найкраще звертатись до методології RAD, коли потрібно швидко розробити ПЗ, вимоги нечітко визначені, а бюджет та терміни обмежені, при цьому потрібна висока гнучкість та адаптивність.

Усі методології розробки програмного забезпечення використовують життєвий цикл розробки систем, який в основному охоплює три основні елементи:

- розуміння проблеми/можливості;
- розробку рішення інформаційних систем;
- впровадження розробленого рішення, навіть якщо спосіб його використання відрізняється.

Наприклад, SDLC будь-якого ІС повинен мати фазу планування та тестування, проте для одного буде застосовано каскадний підхід, коли планування відбувається спочатку, а тестування – в кінці, а для іншого може використовуватися методологія XP і мати невеликі фази планування та тестування, що відбуваються постійно.

Моделі життєвого циклу розробки систем (SDLC) представляють у вигляді поділу на етапи (фази). Існує кілька можливих варіантів структурного поділу життєвого циклу розробки систем (SDLC). Семи етапний життєвий цикл розробки системи включає етапи *планування*, *аналізу*, *проектування*, *розробки*, *тестування*, *впровадження* та фазу *обслуговування*. Кожен етап складається з кількох видів діяльності, або притаманних йому задач, що необхідно вирішити (див. табл. 2.1).

Таблиця 2.1 – Типові задачі для SDLC із семи фаз

Етап	Типові задачі
1. Планування	▪ Визначити та вибрати систему для розробки; ▪ Оцінити доцільність; ▪ Розробити план.
2. Аналіз	▪ Зібрати бізнес-вимоги; ▪ Розробити схеми бізнес-процесів; ▪ Провести порівняльний аналіз витрат на основі купівлі та розробки.
3. Проектування	▪ Запроектувати ІТ-інфраструктуру;

Етап	Типові задачі
	▪ Запроектувати моделі систем.
4. Розробка	▪ Розробити IT-інфраструктуру; ▪ Розробити Бази даних та ПЗ.
5. Тестування	▪ Розробити алгоритми тестів систем; ▪ Виконати тестування систем;
6. Впровадження	▪ Написати детальну документацію користувача системи; ▪ Визначити метод та детальний шлях (план) впровадження системи; ▪ Навчити користувачів системи.
7. Обслуговування	▪ Створити (організувати) службу підтримки користувачів системи; ▪ Провести технічне обслуговування системи; ▪ Забезпечити середовище для підтримки змін в системі.

П’яти етапний життєвий цикл розробки системи включає етапи *дослідження, аналізу, проектування, впровадження та обслуговування* (див. табл. 2.2) із притаманними задачами.

Таблиця 2.2 – Типові задачі для SDLC із п’яти етапів

Етап	Типові задачі
1. Дослідження	▪ Визначити та вибрати систему для розробки; ▪ Оцінити доцільність; ▪ Розробити план управління.
2. Аналіз	▪ Зрозуміти поточну організацію та її поточні системи; ▪ Проаналізувати інформаційні потреби; ▪ Зібрати функціональні вимоги для задоволення інформаційних потреб.
3. Проектування	▪ Розробити (запроектувати) специфікацію на апаратне та програмне забезпечення, персонал, мережу, ресурси даних та пов'язані інформаційні продукти для нової системи; ▪ Розробити логічні моделі нової системи.
4. Впровадження	▪ Придбати (або розробити) програмне та апаратне забезпечення; ▪ Протестувати нову систему та перейти н на нову систему; ▪ Навчити користувачів управляти змінами в системи.

Етап	Типові задачі
5. Обслуговування	▪ Провести огляд після впровадження системи; ▪ Внести необхідні зміни до існуючих систем.

Важливим поняттям в життєвому циклі розробки і впровадженні систем є *процес конверсії*. Цей термін означає перетворення одного стану на інший. Виділяють чотири підходи до конверсії, і кожен з них має свої переваги та недоліки (див. табл. 2.3).

Таблиця 2.3 – Типи конверсії в SDLC

Процес конверсії	Переваги	Недоліки	Ризики
Прямий (Direct) – одночасне вимкнення старих та запуск нових систем.	Швидко і дешево.	Нова система може не працювати	Високий
Паралельний (Parallel) – запуск старої та нової систем одночасно.	Можливість виправити проблеми з новою системою, поки стара працює	Повільніше, ніж прямий підхід, дорого запускати обидві версії, проблеми з продуктивністю через запуск двох систем	Низький
Пілотний (Pilot) – встановлення та тестування нової системи в одній частині організації перед встановленням повсюди.	Пошук та виправлення проблем без впливу на всю організацію	Проблеми з великими обсягами транзакцій можуть не бути виявлені	Помірний
Поетапний (Phased) – система встановлюється послідовно в різних місцях	Потрібно менше персоналу для встановлення, проблеми в одному місці можна виправити перед встановленням в іншому	Різні місця використовують різні версії системи	Помірний

Важливо ретельно вибрати підхід до конверсії, зваживши переваги та недоліки. Також, їх можна комбінувати, наприклад, пілотний підхід у рамках

поетапного підходу; або із застосування паралельного підходу (тобто збереження можливості повернення до старої системи, доки нова система не запрацює безперебійно).

2.3 Управління проектами розробки та впровадження інформаційних систем

Успішне створення та подальший розвиток інформаційних систем визначається високою ефективністю організації управління цього процесу, тобто управління проєктом.

Управління проєктами (англ. project management) – область знань з планування, організації управління та контролю ресурсів з метою успішного досягнення цілей і завдань проєкту в рамках певних обмежень, таких як час, бюджет і якість. Основною метою проектного менеджменту є успішне завершення проекту, яке відповідає вимогам та очікуванням *зацікавлених сторін*.

Проект (англ. Project) – це тимчасова діяльність, спрямована на створення унікального продукту, послуги або результату. Всі проекти мають три ключові характеристики:

- *тимчасовість* – проекти мають визначений логічний початок і кінець;
- *унікальність* – проект створює щось нове: новий продукт, послугу, яка відрізняється від попередніх або результат;
- *системний (поетапний) підхід* – проект проходить через різні фази або стадії, від ініціації до завершення.

Кожен проект має обмеження, в межах яких він повинен бути виконаний. Ці обмеження реалізовані в концепції трикутника проєкту.

Трикутник проєкту (англ. Iron Triangle, залізний трикутник) – це концепція, яка відображає три основні обмеження будь-якого проєкту (Рис. 2.1), що поєднуючись між собою визначають його ефективність та надійність. Обмеження проєкту в межах концепції:

- *Час (Time)* – часові рамки, в яких потрібно завершити проект;
- *Бюджет (Cost)* – фінансові ресурси, виділені на проект;
- *Якість (Quality)* або *обсяг (Scope)* – вимоги та результати, які мають бути досягнуті.

Ці обмеження дають визначення проєкту – як *обсяг* робіт відповідної *якості*, який має бути виконаний в зазначений *час* і за визначений *бюджет*. В ідеальному варіанті, якщо змінити щось одне з цих обмежень, то варто, пропорційно змінити решту обмежень. Аби залізний трикутник завжди залишався рівностороннім, як запорука балансу. Якщо збільшити тільки обсяг, або зменшити тільки час – такий проєкт не матиме ефективних результатів.

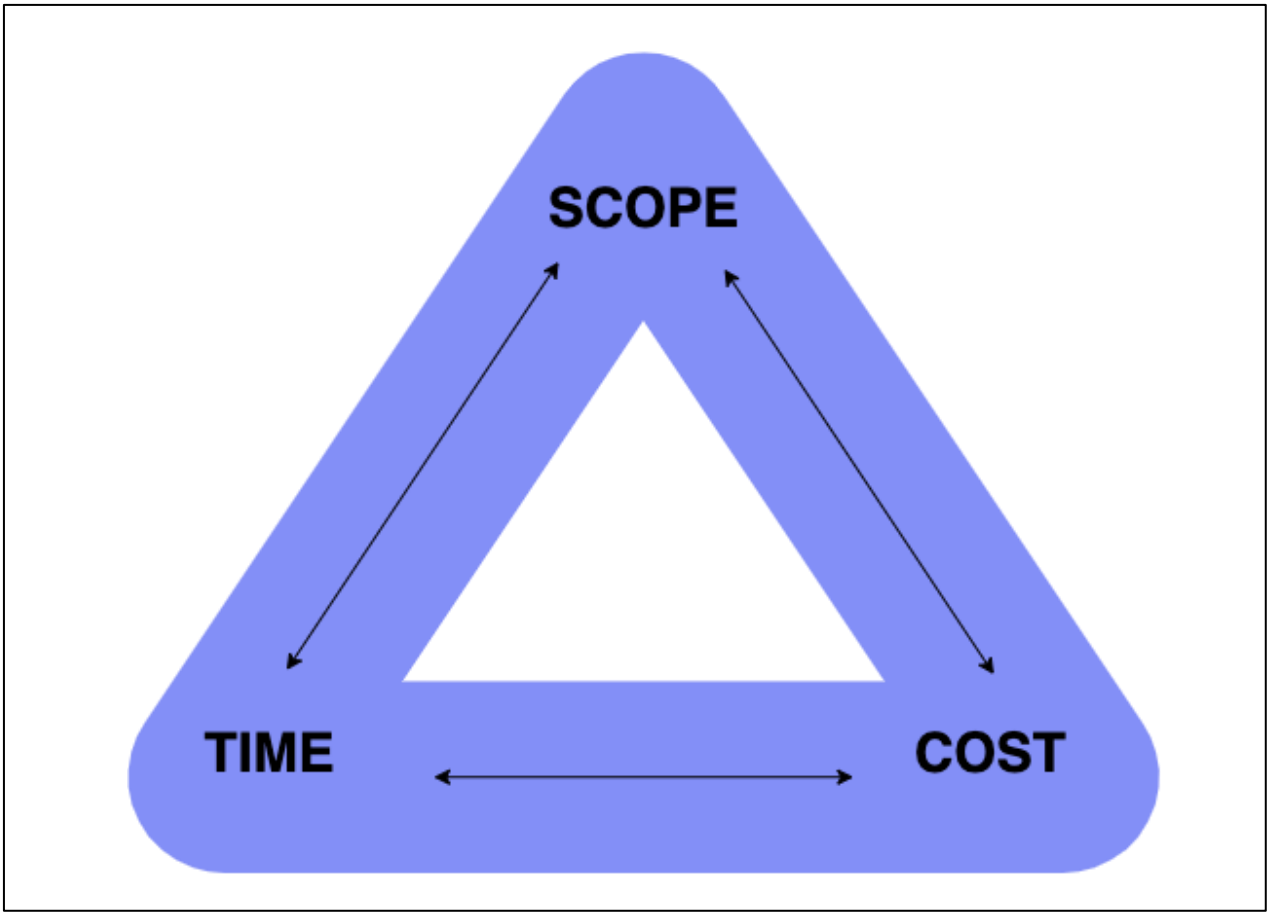

Рисунок 2.1 – Схема концепції «залізний трикутник» або «трикутник проєкту»

Раніше обговорюваний SDLC можна розглядати як специфічний підхід до управління проектами, який є адаптованим підходом до управління проєктами щодо розробки та впровадження інформаційних систем. У багатьох випадках формується проектна команда, що складається як з представників різних відділів, так і з зовнішніх осіб (тобто постачальників та консультантів), особливо для розробки та впровадження великих та складних систем у великих організаціях.

Для команди управління проектами потрібен *керівник проекту* (*проектний менеджер, англ. project manager, PM*). І, хоча, керівник проекту може бути відповідальним за багато завдань, три основні види діяльності є обов'язковими:

- вибір стратегічних проєктів;
- визначення обсягу проєкту (англ. Scope);
- управління ресурсами та підтримку плану проєкту.

Проте, все можна звести до управління ресурсами проєкту, яких, в свою чергу є лише три: люди, час та гроші.

Кілька проєктів об'єднуються в портфелі проєктів (продуктів). Загалом портфель проєктів, яким управляє один менеджер, може містити як пов'язані між собою спільною метою проекти, так і зовсім не пов'язані. Управління програмами – це вищий ієрархічний рівень, на рівні з портфелем, що передбачає

управління групою пов’язаних між собою та взаємозалежних проєктів. Ефективною кількістю проектів в портфелі чи програмі вважають не більше 5 (п’яти). Оскільки, більша кількість знижує ефективність такого управління, змушуючи «розпилювати» увагу, що критично впливає на управління ресурсами та підтримку плану проєктів.

Розробка та впровадження інформаційних систем має кілька успішних стратегій управління проєктами:

- встановлення критеріїв успіху проєкту – визначення показників (метрик) на основі яких буде здійснюватися оцінка якості та успішності;
- *розробка надійного плану проєкту* – детально розроблений план із сприяє зменшенню помилок, а отже ризиків;
- *поділ великих завдань на менші, керовані (декомпозиція)* – визначення структурних компонентів кожної задачі із їх взаємозв’язками;
- *планування змін та забезпечення гнучкості* – принцип конверсії та можливість вносити зміни в проєкт на різних його стадіях;
- *ефективне управління ризиками та проблемами* – визначення несприятливих, загрозливих (негативних) факторів або чинників, що вже впливають на проєкт (проблеми або можуть вплинути на нього у майбутньому (ризики). Управління ризиками та проблемами, в свою чергу, включає такі етапи:
  - *ідентифікація* – визначення потенційних ризиків та проблем, які можуть вплинути на проєкт;
  - *аналіз* – оцінка ймовірності та значення впливу ризиків та проблем на весь проєкт, або його частини та/або часові проміжки такого впливу, з урахуванням умов їх активізації;
  - *розробка стратегії реагування* – визначення методів для уникнення, зменшення або прийняття ризиків;
  - *моніторинг* – регулярний перегляд ризиків і коригування плану.

Процесу управління проєктів властиві помилки, що призводять до небажаних наслідків. Наявність таких помилок також варто передбачати і зважати на їх наслідки. Класичні помилки управління проєктами:

- погана оцінка та/або планування реалізації проекту;
- неефективне управління зацікавленими сторонами;
- недостатнє управління ризиками;
- недостатнє планування;
- низьке забезпечення якості;
- слабкий персонал (відсутність достатніх навичок та досвіду для проекту) та проблеми з командою (тобто труднощі, пов'язані з ефективною роботою

глобальних команд через відсутність особистої взаємодії, різницю в часових поясах, культурні відмінності (включаючи мовні бар'єри);

- недостатнє спонсорство проекту.

Аби нівелювати можливість виникнення цих помилок в управлінні проєктами варто:

- впроваджувати гнучку розробку, що дає змогу швидко реагувати на зміни;
- організувати ефективний план комунікації між учасниками та виконавцями на проєкті, що зменшить ризик помилок;
- застосовувати графік оцінки-конвергенції для показників ефективності проєкту;
- створити офіс управління проєктами, який виконуватимете функцію контролю;
- робота над ретроспективами;
- початок поетапної реалізації;
- управління оцінкою зацікавлених сторін;
- впровадження структури розподілу робіт.

Крім того, також важливо розробити та виконати ефективну стратегію реалізації переваг (особливо для масштабних проектів інформаційних систем, таких як впровадження ERP-систем).

2.4 Управління змінами та ризиками в управлінні проєктами

Всі проєкти в реалізації відрізняються від свого початкового плану. Саме тому гнучкість в їх розробці та своєчасна робота над змінами та ризиками сприяє їх вчасному завершенню.

Масштабні проєкти інформаційних систем мають тенденцію завершуватися із затримкою. Їх графік зривають, запланованого бюджету не вистачає ще й очікуваних результатів досягнуто не в повній мірі. Загалом, серед великих проєктів зі розробки чи впровадження інформаційних систем (бюджет > 15 млн. дол.), в середньому, більш ніж 40% перевищують бюджет, 10% перевищують графік, і 56% з них забезпечують меншу цінність, ніж очікувалося. Основні проблеми, що призводять до більшості невдач таких проєктів: відсутність фокусу (тобто нечіткі цілі, відсутність бізнес-фокусу); проблеми зі змістом (тобто зміна вимог, технічна складність); проблеми з навичками (неузгоджена команда, брак навичок); та проблеми з виконанням (нереалістичний графік, реактивне планування).

Крім того, варто зважати на пропозиції щодо успіху проекту: управління стратегією та зацікавленими сторонами (з урахуванням таких факторів, як: чіткі цілі, чітко визначений бізнес-кейс, узгодженість основного обсягу проекту, мінімізований та стабільний обсяг проекту, надійні контракти з постачальниками з чіткими обов'язками, підтримка керівництва); оволодіння технологіями та

контентом (з урахуванням таких факторів, як стандартизовані та перевірені технології програмного забезпечення, залучення користувачів до формування рішення); формування команди та можливостей (з урахуванням таких факторів, як досвідчений керівник проекту, кваліфікована та мотивована команда проекту, стійке поєднання внутрішніх та зовнішніх ресурсів); та досягнення успіху в практиці управління проектами (з урахуванням таких факторів, як надійні оцінки та плани, належна прозорість щодо статусу проекту, перевірені методології та інструменти).

Впровадження нових систем в організації неминуче призведе до опору з боку користувачів існуючих чи застарілих систем. Це цілком нормально, оскільки людська природа схильна ставитися до всього нового з підозрою. Дуже часто люди ставлять питання: «Що я з цього отримаю?». Як наслідок, ще один важливий аспект це управління змінами.

Управління змінами (англ. Change Management) – це набір методів для переходу систем (також застосовується для окремих осіб груп, організацій, тощо) з поточного стану в майбутній із очікуваним результатом. Фактично, підхід до еволюції та управлінні політикою проектування та впровадження системи.

Ефективне управління змінами допомагає компаніям зменшити ризики та витрати. Найскладнішим аспектом управління змінами є робота з людьми (як у комунікаційному, так і в культурному аспектах).

Організації повинні переконатися, що люди розуміють переваги змін як для організації, так і для них самих, а також ризики відсутності змін. Водночас необхідно піклуватися про потреби людей, такими принципами:

- Зосередження уваги на вищому керівництві та розвитку лідерства: лідери повинні бути ініціаторами змін, очолювати управління змінами, брати на себе відповідальність за управління змінами та подавати хороші приклади для наслідування;
- Створення бачення змін: розуміння стратегічного бачення, створення переконливої історії та забезпечення комплексності та оперативності бачення;
- Визначення стратегії змін: оцінка готовності до змін, вибір найкращої конфігурації змін та впровадження управління змінами;
- Залучення людей з різних підрозділів та різних рівнів організації: краще розуміння та краща освіта щодо змін будуть спрощені;
- Формування та підтримка відданості: формування команд, управління зацікавленими сторонами, ефективна та достатня комунікація, подолання опору, поширення передового досвіду та знань/навичок, а також визнання досягнень у кожній можливій публічній можливості;
- Управління ефективністю окремих осіб: організації повинні чітко та чесно повідомляти працівникам про те, що від них очікується та як це буде

вимірюватися. А після встановлення показників успіху та невдачі, а також стимулів/штрафів, їх слід належним чином дотримуватися. Організаціям також необхідно перерозподілити певні ролі, щоб забезпечити дотримання змін та нового підходу/процесу людьми;

- Реалізація бізнес-вигод: побудова бізнес-кейсу та постійна кількісна/відчутна оцінка переваг;
- Розробка необхідної культури: визначення розриву в культурі та впровадження необхідних культурних змін;
- Коригування структури: визначення та впровадження необхідних змін в організаційній структурі для впровадження змін;
- Підготовка до неочікуваного: організаціям необхідно постійно оцінювати хід та ефективність програми управління змінами та своєчасно ефективно реагувати на неочікувані події. Крім того, їм необхідно мати заходи щодо зменшення ризиків, щоб забезпечити мінімальні збої в бізнесі під час процесу управління змінами.

Ще одним важливим фактором, що впливає на успіх розробки інформаційних систем, є ризики проекту. *Ризики* – це невизначені події або умови, що можуть відбутися в майбутньому, і матимуть негативний вплив на результати проекту.

Ризики можуть виходити зсередини (тобто припинення підтримки з боку вищого керівництва, недостатність виділених ресурсів, раптовий відхід ключових людей з проекту) або ззовні (тобто зміни в галузевих стандартах, нові правила та переміщення/переваги конкурентів у подібних/тих самих проектах) організації. Як і у випадку з будь-якими іншими типами ризиків, слід запровадити ефективний процес (систему) управління ризиками проекту:

- визначення ризику;
- кількісна оцінка ризиків (ймовірних та впливу);
- розробка та впровадження рішень з управління ризиками;
- постійна оцінка ефективності впроваджених рішень та внесення необхідних коригувань/змін.

Тим часом ефективні практики управління ризиками не є поширеним явищем для багатьох організацій, і лише в останнє десятиліття управління ризиками стало серйозно розглядатися як частина управлінських обов'язків.

Існує кілька різних варіантів створення нових інформаційних систем:

- *внутрішня розробка* – організація володіє достатньою кількістю ресурсів (працівники відповідної кваліфікації) для самостійної розробки та впровадження інформаційної системи (реалізації проекту);
- *зовнішня розробка (аутсорсинг, англ. outsourcing)* – організація залучає зовнішні ресурси для розробки та впровадження інформаційної системи (реалізації проекту);

- *купівля (англ. procurement)* – купівля готового продукту (інформаційної системи). Як правило, являє собою проведення торгів на конкурсній основі, відповідно до внутрішніх політик організації;
- *постачальник послуг додатків (англ. Application Service Providers, ASP)* – використання програмного продукту, що належить постачальнику, за підпискою (ліцензією), шляхом доступу через мережу інтернет (замість встановлення на робочих комп'ютерах). При цьому немає потреби турбуватись про інфраструктуру, обслуговування та оновлення.

Кожен з цих варіантів має свої переваги та недоліки. Внутрішня розробка дає повний контроль над кінцевою системою (або результатом). При цьому команда, що розробила та підтримує продукт постійно нарощує свої технічні навички та функціональні знання. Водночас, такий варіант вимагає цілеспрямованих зусиль персоналу, а розробка може бути повільною, що збільшить витрати.

Аутсорсинг – це економія коштів, легкість застосування нових технології. Такий варіант вигідний тим, що команда вже сформована та має досвід і навички для ефективного управління персоналом інформаційних систем. Серед недоліків варто відмітити можливу втрату контролю над розробкою та залежність від постачальника (аутсорсера) в майбутньому.

Купівля – це найшвидший підхід з усіх. Компанія після процесу інтеграції та впровадження може приступати до повноцінного використання системи. А за невелику кількість ітерацій вирішуються проблеми, що можуть виникнути. Функціональність такої придбаної системи доводиться прийняти, бо вже не підлягає модифікації, що і є недоліком.

Використання постачальників послуг додатків (ASP) перш за все вигідні своєю невеликою вартістю, оскільки дає можливість зекономити на внутрішній інфраструктурі, початковому капіталі та непотрібно тримати спеціалізований внутрішній персонал з інформаційних систем. Деяким компаніям легше «орендувати» програмне забезпечення у ASP та уникнути проблем, пов’язаних із встановленням, експлуатацією та обслуговуванням складних систем, таких як CRM, ERP. Додатки від ASP дозволяють швидше реагувати на ринок завдяки .

ASP дає менший контроль над додатками, при цьому є досить універсальними, тобто дозволяє приблизно 20% налаштувань для будь-якої компанії та вирішують рутинні задачі.

Не варто вкладатись в додаткову розробку ASP, оскільки витрати на програмне забезпечення не створюю додаткової вартості для компанії, на відміну від всіх попередніх підходів.

Організації можуть вибрати будь-який із чотирьох підходів до розробки після оцінки їх переваг та недоліків до розробки та врахування своїх організаційних обставин та вимог. Також, для масштабних портфелів можна застосувати різні підходи одночасно: деякі проєкти віддавати на аутсорсинг, інші створювати внутрішньо або комбінувати із покупкою та ASP.

2.5 Стандарти управління проєктами

Стандарти управління проектами встановлюють загальні принципи, регламентують процеси й описують кращі підходи та практики, які допомагають структурувати роботу над проєктом.

Вони створені для забезпечення узгодженості, якості та ефективності управління проектами в різних галузях (не тільки при розробці інформаційних систем).

ДСТУ ISO 21500:2022 є українською версією міжнародного стандарту ISO 21500 (серії стандартів), що надає рекомендації щодо управління проектами, адаптовані для національних умов. Величезною перевагою такої серії стандартів є міжнародна сумісність, оскільки, це адаптована версія міжнародного стандарту, що забезпечує відповідність міжнародним вимогам. Ці стандарти надають загальні вказівки, які можуть бути адаптовані під різні типи проектів.

ISO 21500 «Управління проектами, програмами та портфелями. Контекст та концепції» – визначає основні принципи та концепції управління проектами, програмами та портфелями, що створює основу для розробки ефективних підходів до управління. Він допомагає забезпечити успішне виконання проектів у відповідності до міжнародних норм. Є українським стандартом, що перекладений з міжнародного ISO 21500:2021. Також це ціла серія стандартів (ISO 21500), що містить перелік документів (стандартів), які регламентують різні аспекти галузі управління проєктів.

ISO 21502 «Управління проектами, програмами та портфелями. Настанови щодо управління проектами» – надає настанови щодо управління проектами, включаючи процеси, методи та інструменти. Його цінність полягає в наданні структурованого підходу до управління проектами, що підвищує ефективність і знижує ризики невдачі.

ISO 21503 «Управління проектами, програмами та портфелями. Настанови щодо управління програмами» – надає рекомендації щодо управління програмами, що включають кілька взаємопов'язаних проектів. Він забезпечує оптимізацію ресурсів та покращення координації між проектами для досягнення стратегічних цілей організації.

ISO 21504 «Управління проектами, програмами та портфелями. Настанови щодо управління портфелями» – охоплює управління портфелями, що включають управління декількома проектами та програмами одночасно. Він допомагає організаціям досягати своїх стратегічних цілей через ефективне управління та пріоритизацію проектів.

ISO 21505 «Управління проектами, програмами та портфелями. Настанови щодо врядування» – визначає принципи та підходи до врядування проектів, програм та портфелів. Він надає рамки для забезпечення прозорості, відповідальності та належного нагляду за проектною діяльністю, що сприяє досягненню цілей і мінімізації ризиків.

ISO 21508 «Управління здобутою цінністю в управлінні проектами та програмами» – охоплює управління здобутою цінністю (англ. Earned Value Management, EVM) у проектному менеджменті, що дозволяє оцінювати прогрес проекту в термінах вартості та часу. Його використання допомагає проєктним командам контролювати відхилення та приймати обґрунтовані рішення.

ISO 21511 «Ієрархічна структура робіт для управління проектами та програмами» – описує методи створення ієрархічної структури робіт (англ. work breakdown structure, WBS), яка допомагає організовувати завдання та ресурси у проекті. Використання WBS дозволяє чітко визначати обсяг робіт і забезпечувати ефективний контроль над виконанням завдань. WBS орієнтована на декомпозицією проєкту на менші частки.

У 1957 році Міністерство Оборони США запровадила Техніку Оцінювання та Аналізу Програм (англ. Program Evaluation and Review Technique, PERT), в основу якої була покладена концепція структури декомпозиції робіт. PERT був запропонований ВМС США для підтримки розробки ракетної програми «Поларіс».

Серед недоліків, варто відмітити, що стандарт групи ISO 21500 може не містити достатньо деталей для специфічних випадків. А також, він обмежений і може вимагати додаткового контексту для повного розуміння та застосування в специфічних умовах.

ISO 21500 є міжнародним стандартом, який надає загальні вказівки щодо принципів і термінології управління проектами. Цей стандарт забезпечує високий рівень опису основних концепцій та термінів управління проектами.

У 1987 році, Інститут Проєктного Менеджменту (англ. Project Management Institute, PMI) задокументував розширення цих технік WBS та PERT на використання не лише в системах оборони. І в 1996 році в книзі Настанова Знань по Проєктному Менеджменту (англ. A Guide to the **P**roject **M**anagement **B**ody **O**f **K**nowledge, PMBOK) виклав концепт WBS, та PERT але для більш загального використання.

PMBOK® Guide – це зібрання знань з управління проектами, розроблене Інститутом управління проектами (PMI), часто в літературі називається стандартом. Проте, PMBOK надає детальні інструкції та найкращі практики для управління проєктами. Він охоплює всі аспекти проектного менеджменту та допомагає проектним менеджерам ефективно планувати, виконувати та завершувати проекти, зменшуючи ризики невдачі.

PMBOK є основою для сімейства міжнародного стандарту ISO 21500 та Американського національного стандарту ANSI/PMI 99-001-2008. Ця книга найбільш розповсюджене зібрання знань з управління проєктами, що використовується у всьому світі. Також, сертифікація проєктних менеджерів PMP (англ. Project Management Professional) відбувається на основі PMBOK.

PMBOK має на меті неухильне виконання дев'яти функцій, що поділяються на дві групи – *основні* (спрямовані на управління цілями проєкту) та *додаткові* (спрямовані на управління певними об'єктами проєкту). Функції PMBOK:

- **Основні**:
  - *Управління обсягом проєкту* – контролює проєкт за визначення і дотримання його мети, завдань і цілей.
  - *Управління витратами* – контролює фінанси проєкту завдяки накопиченню, аналізу та складанню звітів по витратах.
  - *Управління часом* – контролює вчасне виконання задач проєкту, передбачає розробку календарних графіків та планів.
  - *Управління якістю* – контролює належний рівень якості виконання задач проєкту, відповідно до стандартів, що встановлені для проєкту.
- **Додаткові:**
  - *Управління людськими ресурсами* – контролює і координує діяльність людей, залучених до проєкту.
  - *Управління комунікаціями* – контролює та накопичує інформацію, якою обмінюються члени проєктної команди, для успішного завершення проєкту.
  - *Управління контрактами/постачанням* – контролює процеси відбору, переговорів і підписання договорів на замовлення матеріалів, устаткування та послуг, необхідних для проєкту.
  - *Управління ризиком* – залежить від ступеня невизначеності проєкту і базується на знаннях та досвіді із зазначенням умов реалізації конкретного проєкту.
  - *Управління проєктною інтеграцією* – контролює забезпечення належну координацію всіх функцій проєкту.

Загалом, PMBOK ілюструє концепцію та філософію підходу до управління проєктами. На прикладі двох останніх версій цього документу (7-е видання, опубліковане в 2020 році та 6-е видання, опубліковане в 2017 році) можна порівняти як змінювалось бачення управління проєктів, та які тенденції переважають в цій галузі.

6-е видання PMBOK охоплює 49 процесів, розподілених по п’яти групах процесів і десяти областях знань (див. рис. 2.2).

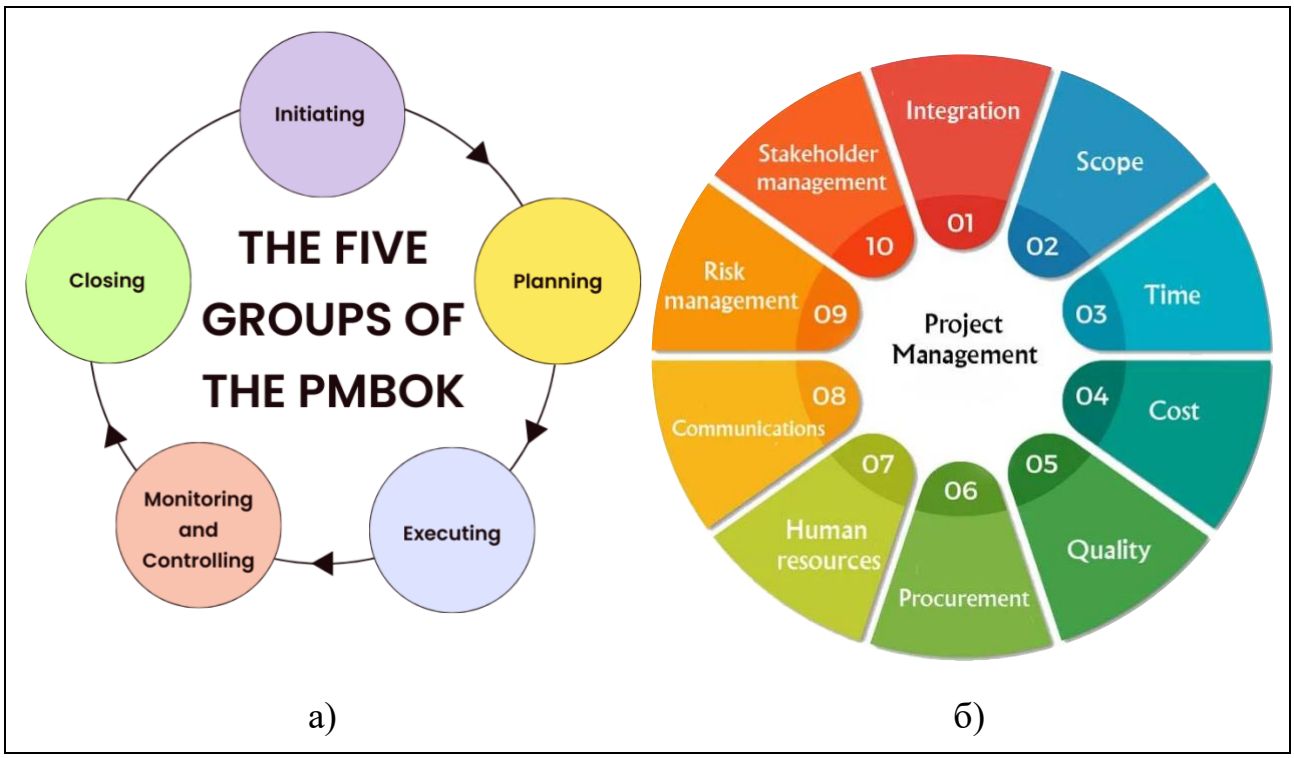

Рисунок 2.2 – Схема концепції PMBOK® Guide 6-е видання:

а) - групи процесів; б) - області знань

Дослідимо 10 областей знань в PMBOK 6-му виданні (рис. 2.2.*б*) та 49 процесів, розподілених між ними:

1. **Управління інтеграцією проекту (англ. Project Integration Management):** область знань, яка визначає сукупність процесів, що об'єднують і координують різні його частини щоб забезпечити успішного завершення проєкту. Ця область знань відповідає за координацію всіх елементів проєкту, як диригент. Їй властива розробка статуту проєкту (англ. project charter, ще відомий як «паспорт проєкту») – документ, розроблений ініціатором який авторизує існування проєкту, надає повноваження, визначає ресурси. Також, цій області знань притаманні процеси: розробка плану управління проєктом, управління виконанням, моніторинг та контроль роботи, контроль змін, закриття проекту.
2. **Управління обсягом проекту (англ. Project Scope Management):** область знань, яка визначає процеси, що забезпечують виконання проекту в рамках його обмежень, контролюючи обсяг робіт, необхідних для успішного завершення проекту. Вона охоплює процеси для визначення того, що буде включено, а що не буде включено в проект, на що результат проєкту матиме вплив, на чому не позначиться. Також, цій області знань притаманні процеси: планування управління обсягом, збирання вимог, визначення обсягу, створення WBS, підтвердження та контроль обсягу.
3. **Управління часом проекту (англ. Project Schedule Management):** область знань, яка визначає процеси, що забезпечують своєчасне завершення

проекту. Вона включає визначення операцій та їх оцінку, визначення їх послідовності, оцінку ресурсів і тривалості, розробку та управління планом проекту. Також, цій області знань притаманні процеси: планування управління розкладом, визначення дій, послідовності дій, оцінка ресурсів, оцінка тривалості, розробка та контроль розкладу.

4. **Управління вартістю проекту (англ. Project Cost Management):** область знань, яка визначає процеси планування, оцінки, бюджетування, фінансування, управління та контролю витрат, щоб проект був завершений у межах затвердженого бюджету. Вона включає в себе декілька ключових етапів, таких як планування вартості, оцінка вартості, складання бюджету, контроль витрат, та аналіз відхилень. Також, цій області знань притаманні процеси: планування управління вартістю, оцінка вартості, визначення бюджету, контроль вартості.
5. **Управління якістю проекту (англ. Project Quality Management):** область знань, яка визначає комплекс процесів, спрямованих на забезпечення відповідності якості проекту встановленим стандартам та очікуванням замовника. Вона включає в себе планування, забезпечення, контроль та покращення якості проєкта. Також, цій області знань притаманні процеси: планування управління якістю, управління якістю та контроль якості.
6. **Управління ресурсами проекту (англ. Project Resource Management):** область знань, яка визначає комплекс процесів планування, організації, моніторингу та контролю ресурсів, необхідних для успішного виконання проекту. Вона об'єднує людські ресурси, обладнання, матеріали, бюджет та час. Ефективне управління ресурсами є критично важливим для досягнення цілей проекту. Також, цій області знань притаманні процеси: планування управління ресурсами, оцінка ресурсів, набір команди, розвиток команди, управління командою, контроль ресурсів.
7. **Управління комунікаціями проекту (англ. Project Communications Management):** область знань, яка визначає комплекс процесів планування, організації, моніторингу та контролю інформаційного обміну між учасниками проекту та зацікавленими сторонами. Головною метою є забезпечити ефективне накопичення, обробку та обмін інформації, необхідної для успішної реалізації проекту. Також, цій області знань притаманні процеси: планування управління комунікаціями, управління комунікаціями та моніторинг комунікацій.
8. **Управління ризиками проекту (англ. Project Risk Management):** область знань, яка визначає комплекс систематичних процесів виявлення, аналізу, оцінки, планування та контролю ризиків, які можуть вплинути на успішне виконання проекту. Головною метою є мінімізувати негативні наслідки можливих ризиків, шляхом розробки планів дій, щодо їх усунення та/або їх наслідків для досягнення цілей проекту, в рамках встановлених термінів, бюджету та якості. Також, цій області знань притаманні процеси: планування управління ризиками, ідентифікація ризиків, якісний аналіз, кількісний

аналіз, планування реагування на ризики, впровадження реагування на ризики, моніторинг ризиків.

9. **Управління закупівлями проекту (англ. Project Procurement Management):** область знань, яка визначає комплекс процесів управління зовнішніми закупівлями, необхідними для виконання проекту. Також, цій області знань притаманні процеси: планування управління закупівлями, проведення закупівель та контроль закупівель.
10. **Управління зацікавленими сторонами проекту (англ. Project Stakeholder Management):** область знань, яка визначає комплекс процесів виявлення, аналізу та залучення зацікавлених сторін проєкту для забезпечення їх підтримки та успішного виконання проекту. Важливо виявити всіх осіб, груп або організацій, які можуть впливати на проект та є зацікавленими в успішному реалізації проекту, та розробку стратегій для ефективного управління їх очікуваннями та потребами. Також, цій області знань притаманні процеси: ідентифікація зацікавлених сторін, планування залучення, управління залученням, моніторинг залучення.

Розглянемо 5 груп (фаз або стадій проєкту) та притаманні їм процеси (рис. 2.2.*а*):

- *Група процесів ініціації* (англ. Initiation) – комплекс процесів, що сприяють формальній авторизації початку нового проєкту. Ці процеси ініціації часто виконуються поза рамками проєкту, тобто перед його початком, в ході яких, визначають початкові параметри проєкту (документується вихідні припущення, обмеження, призначається менеджер). До групи процесів ініціації належать такі процеси:
  - Визначення залучених сторін та (відповідно до їх вимог) розробка попереднього опису змісту проєкту;
  - Розробка Статуту проєкту (Project Charter).
- *Група процесів планування* (англ. Planning) – комплекс процесів, що визначають, уточнюють і планують дії, необхідні для досягнення цілей проєкту. До групи процесів планування належать такі процеси:
  - Розробка плану управління проєктом;
  - Планування змісту згідно вимог (замовника зовнішнього або власного);
  - Визначення обсягу;
  - Створення ієрархічної структури робіт (WBS);
  - Визначення складу операцій;
  - Визначення взаємозв'язків операцій;
  - Оцінка ресурсів операцій;
  - Оцінка тривалості операцій;
  - Розробка розкладу;

- Вартісна оцінка;
- Розробка бюджету витрат;
- Планування якості;
- Планування людських ресурсів;
- Планування комунікацій;
- Планування управління ризиками;
- Ідентифікація ризиків;
- Якісний аналіз ризиків;
- Кількісний аналіз ризиків;
- Планування реагування на ризики;
- Планування закупівель;
- Планування контрактів.

- *Група процесів виконання* (англ. Execution) – комплекс процесів, що об'єднує людські та інші ресурси, передбачені проєктом, для виконання плану управління даного проєкту. До групи процесів виконання входять такі процеси:
  - Керівництво та управління виконанням проєкту;
  - Процес забезпечення якості;
  - Набір команди проєкту;
  - Розвиток команди проєкту;
  - Поширення інформації;
  - Запит інформації у продавців;
  - Вибір продавців.
- *Група процесів моніторингу та контролю* (англ. Monitoring & Control) – комплекс процесів, що спрямовані на регулярну оцінку прогресу проєкту, щоб виявити відхилення від плану управління проєктом, і, в разі необхідності, провести коригувальні дії для досягнення цілей проєкту. До групи процесів моніторингу і управління входять такі процеси:
  - Моніторинг та управління роботами проєкту;
  - Загальне управління змінами;
  - Підтвердження обсягу (відповідно змісту та цілям);
  - Управління обсягом;
  - Управління розкладом;
  - Управління вартістю;
  - Процес контролю якості;

- Управління командою проєкту;
- Звітність по виконанню;
- Управління учасниками проєкту;
- Спостереження і управління ризиками;
- Адміністрування контрактів.

- *Група завершальних процесів* (англ. Closure) – комплекс процесів, що формалізує приймання продукту, послуги або результату і підводить проєкт або фазу проєкту до правильного завершення. Група завершальних процесів містить такі процеси:
  - Закриття проєкту;
  - Закриття контрактів.

Життєвий цикл проєкту на основі процесного підходу, представленого в PMBOK 6-го видання, можна відобразити за допомогою лінійної схеми із чотирьох етапів (див. рис. 2.3).

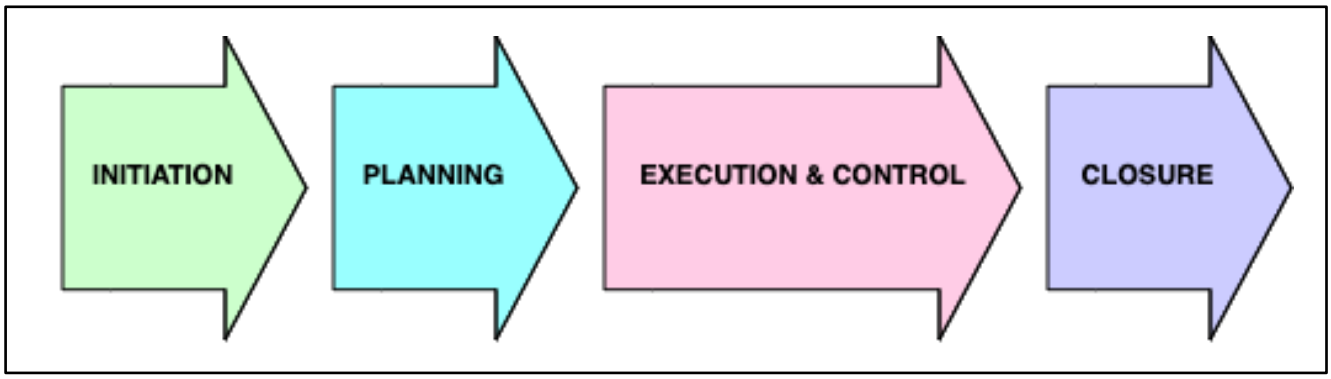

Рисунок 2.3 – Схема життєвого циклу проєкту на основі PMBOK® Guide (6-е видання)

Як бачимо, фази *«Моніторинг та контроль»* та *«Виконання»* об'єднані в єдиний етап. Все це тому, що вони завжди відбуваються паралельно. Всі активності, що були заплановані на проєкті, виконання яких створює цінність (результати проєкту), потребують систематичного моніторингу.

7-е видання PMBOK® Guide, опубліковане в 2020 році, є фінальною версією, яка робить акцент в сторону більш гнучкої моделі управління проєктами. Як результат відходить від процесного підходу, і натомість, визначає 12 принципів управління проєктами. А замість області знань, визначає 8 доменів (доменних зон – сфер виконання проєкту) його продуктивності (див. рис. 2.4).

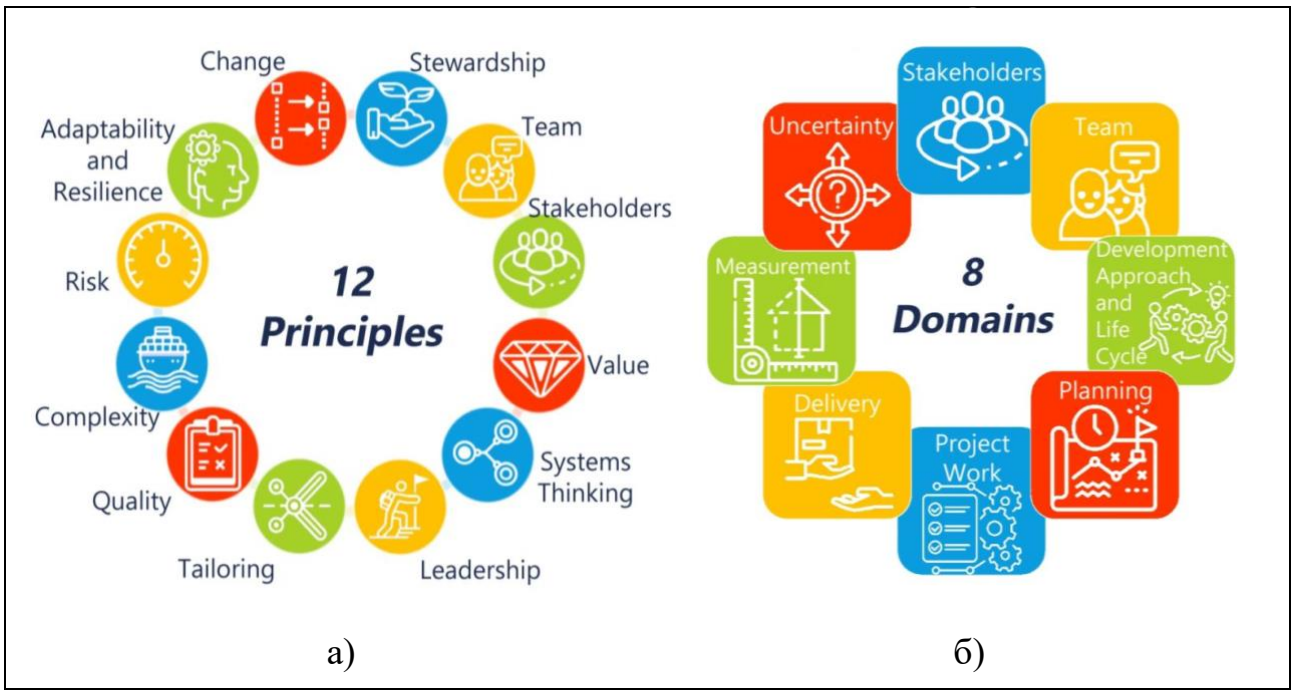

Рисунок 2.4 – Схема концепції PMBOK® Guide 7-е видання:

а) - принципи управління; б) - доменні зони

Розглянемо 12 принципів в PMBOK 7-го видання:

1. **Відповідальне керівництво (англ. Stewardship):** це підхід до управління, що заснований на етичних принципах, нормами та відповідальності, із постійною практикою піклування про ресурси, що знаходяться у розпорядженні.
2. **Команда (англ. Team):** підхід до управління, що базується на побудові ефективної команди проєкту, що зосереджена на досягненні цілей проєкту.
3. **Орієнтація на зацікавленні сторони (англ. Stakeholder Focus):** стратегічний підхід, який передбачає визначення та пріоритизацію потреб всіх сторін, що мають вплив на діяльність організації або проєкту (клієнти, колеги, відділи, організації, тощо). Це включає в себе не лише клієнтів, але й співробітників, партнерів, постачальників, членів громади та інших зацікавлених осіб. Метою є ідентифікація та подальше зосередження на задоволенні потреб зацікавлених сторін.
4. **Поставка цінності (англ. Value Delivery):** підхід, що заснований на меті створити та доставити цінність (продукти або послуги) для клієнтів та організації, щоб задовольнити їхні вимоги та очікування. Ефективна доставка цінності означає не просто доставку продукту або послуги, а й забезпечення того, щоб вона вирішувала проблему, задовольняла потребу або покращувала досвід клієнта значущим чином.
5. **Системне мислення (англ. Systems thinking):** підхід, який розглядає об'єкти як взаємопов'язані частини цілої системи, а не як окремі незалежні елементи.

І зосереджується на прийнятті управлінських рішень на основі аналізу даних та ризиків, щоб зрозуміти, як система працює в цілому.

6. **Лідерство (англ. Leadership):** підхід, що базується взаємодії із командою, зацікавленими сторонами чи клієнтами на основі лідерства в контексті управління проєкту. Лідерство включає в себе такі якості, як надійність, гуманність, мужність та дисципліна, а також здатність до адаптації та комунікації. Це не те саме, що бути начальником.
7. **Адаптація (англ. Tailoring):** підхід, що заснований на адаптації та налаштування процесів та інструментів управління проектами відповідно до унікальних характеристик конкретного проекту, його оточення та потреб зацікавлених сторін. Це передбачає вибір та модифікацію частин PMBOK, які найбільше підходять для даного проекту, щоб забезпечити його успішне виконання та досягнення цілей.
8. **Якість (англ. Quality):** підхід, за якого ключовим аспектом успішного виконання проєкту є відповідність потребам зацікавлених сторін та процесам прийняття, що гарантує, що процеси проєкту є належними та відповідають очікуванням.
9. **Складність (англ. Complexity):** підхід, який підкреслює необхідність регулярної оцінки та управління умовами або подіями, що можуть впливати на проєкт. Складність не можна контролювати, але проєктні команди можуть адаптувати свої дії, щоб зменшити її вплив, а також повинні уважно виявляти та пом'якшувати фактори складності.
10. **Можливості та Ризики (англ. Opportunity & Risk):** підхід, що визначає необхідність використовувати можливості та ефективно управляти ризиками для покращення результатів проєкту.
11. **Адаптивність та Життєстійкість (англ. Adaptability & Resilience):** підхід, що визначає здатність проєкту та команди реагувати на зміни та невизначеності, гнучко адаптуючись до нових умов. Це передбачає здатність швидко переходити від однієї стратегії до іншої, враховувати нові обставини та використовувати їх для досягнення цілей проєкту.
12. **Зміни (англ. Change):** підхід, що стосуються здатності адаптуватися до змін, що виникають під час виконання проєкту. Це включає в себе управління запитами на зміни, оцінку їх впливу та впровадження схвалених змін..

А також 8 сфер виконання проекту (доменів):

1. **Зацікавлені сторони (англ. Stakeholders):** Управління та залучення зацікавлених сторін для забезпечення підтримки проекту.
2. **Команда (англ. Team):** Управління командою проекту, розвиток та підтримка її ефективності.
3. **Підходи до розвитку та життєвий цикл проекту (англ. Development Approaches & Life Cycle):** Вибір підходу та структури життєвого циклу для виконання проекту.

4. **Планування проекту (англ. Planning):** Створення та управління планом проекту для досягнення його цілей.
5. **Робота над проектом (англ. Project Work):** Управління виконанням проекту, контроль роботи.
6. **Доставка результатів (англ. Delivery):** Забезпечення своєчасної доставки результатів проекту.
7. **Вимірювання ефективності (англ. Measurement):** Оцінка прогресу та результатів проекту для забезпечення відповідності запланованим цілям.
8. **Невизначеність та ризики (англ. Uncertainty & Risk):** Управління ризиками та невизначеністю для забезпечення стійкості проекту.

Робота у сферах виконання проекту (доменах) ґрунтується на принципах управління проектами. Незважаючі на концептуальний збіг між принципами та сферами виконання, принципи визначають поведінку, а сфери виконання відображають напрями, де можно застосувати цю поведінку. Схематично в стандарті це зображено на рис. 2.5.

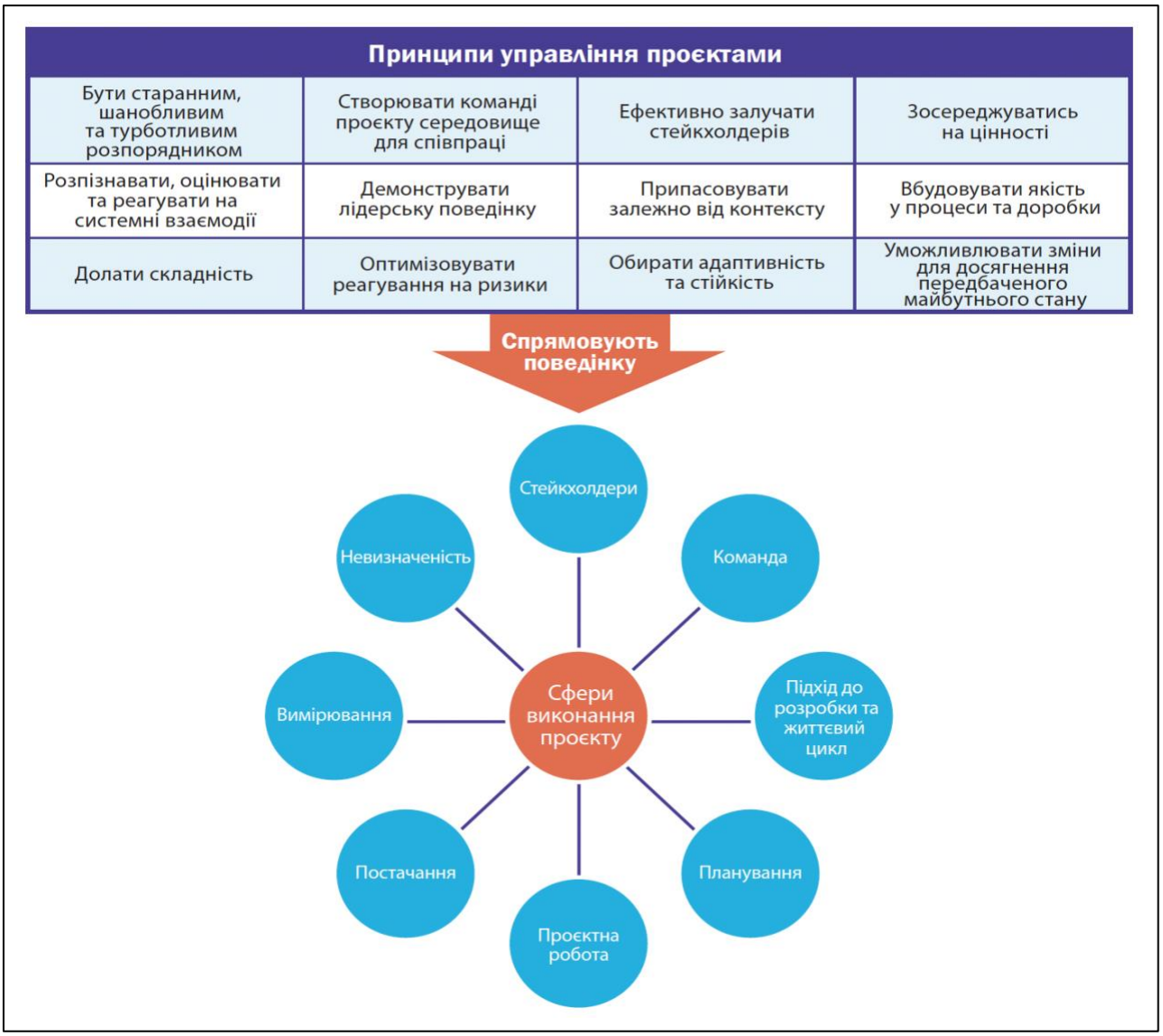

Рисунок 2.5 – Взаємозв’язок принципів та доменів проєкту в PMBOK.

Отже, основні відмінності між 6-м та 7-м виданням PMBOK полягає в структурному підході: 6-е видання зосереджене на процесному підході (49 процесів, згрупованих у 10 областей знань, що розподілені по п'яти групах процесів), а 7-е видання орієнтоване на принципи (12 принципів управління проектами та 8 доменів продуктивності проекту). Такий підхід відображають ширший погляд на управління проектами, з акцентом на цінності й результат. Він відмовляється від традиційних областей знань та процесів і ставить великий акцент на гнучкості та адаптивності, підтримуючи використання будь-якої моделі, яка найбільш підходить для конкретного проекту, включаючи гнучкі, каскадні та гібридні підходи.

PMBOK 6 слідує за традиційною лінійною структурою (waterfall), в якій проєкти проходять через послідовні фази (хоча згадує agile), а PMBOK 7 відкритий до будь-якого підходу, включаючи як послідовні, так і гнучкі методи, з особливим акцентом на цінність і результати.

PMBOK 7 – це значний крок вперед у проектному менеджменті, який відображає сучасні тенденції та вимоги ринку. Він надає керівникам проектів більшу свободу у виборі підходів та методів, орієнтуючись на результати та цінності. У той же час PMBOK 6 залишається актуальним для тих, хто працює з більш традиційними проектами, де необхідна чітка структура і контроль. Вибір між PMBOK 6 та PMBOK 7 залежить від типу проекту та контексту його реалізації.

Кожен із розглянутих вище стандартів має свої унікальні особливості та підходить для різних типів організацій та проектів. PMBOK пропонує комплексний підхід до управління проектами з акцентом на процеси, ISO 21500 орієнтований на інтеграцію з іншими стандартами та забезпечення якості, фокусується на компетенціях керівника проекту, що робить його особливо корисним для розвитку.

2.6 Методології та фреймворки управління проєктами

Більшість задач в нашому житті є повторюваними або однотипними, що з одного боку, складає рутину життя, а з іншого визначаю шаблони (патерни) поведінки. Якщо відійти від буденності людського життя, то виявляється (див. п. 2.2), буденність життя проєкту також має однотипні повторювані дії.

Фреймворки та методології допомагають командам організувати роботу з гнучкістю та ефективністю, сприяючи більш швидкому та узгодженому виконанню завдань. Вони забезпечують структуру, методи та інструменти для успішної реалізації проєктів і дозволяють уніфікувати процеси, полегшують комунікацію в команді та допомагають досягати поставлених цілей. Загалом, вони визначають збір правил, принципів, шаблонів, алгоритмів, застосування яких якісно характеризує успіх проєкту.

Методологія (англ. Methodology) – це системий підхід до виконання певного виду діяльності або досягнення результату, який включає набір принципів, правил, процесів та технік. В контексті проектного менеджменту, методологія –

це детально визначений спосіб управління проєктами, що охоплює всі етапи проекту, від його початку до завершення, з чіткими інструкціями щодо виконання кожного етапу. Основні характеристики методології:

- *Структура*: Методологія має чітко структурований підхід з етапами та задачами.
- *Регламентація*: Визначає, які процеси і в якій послідовності мають бути виконані для досягнення мети.
- *Універсальність*: Може застосовуватись до різних проектів з певною адаптацією.
- *Контроль і вимірювання*: Методології включають механізми для моніторингу прогресу та оцінки результатів.

Фреймворк (англ. Framework) – це структурована система, набір методів, інструментів і правил, які надають базову структуру для організації процесів, розробки програмного забезпечення або управління проектами. Фреймворк дозволяє організувати та стандартизувати роботу команд, щоб забезпечити ефективне виконання завдань, забезпечити гнучкість та узгодженість між етапами процесу. Основні характеристики фреймворку:

- *Базова структура:* фреймворк надає загальні принципи та рамки для організації роботи, але залишає простір для адаптації залежно від потреб конкретного проекту.
- *Повторюваність:* фреймворк визначає процеси та методи, які можна використовувати багаторазово, незалежно від типу проекту чи завдання.
- *Модульність:* фреймворки часто складаються з окремих модулів або компонентів, які можна використовувати незалежно або разом, відповідно до вимог проекту.
- *Гнучкість:* фреймворк дозволяє команді адаптувати процеси до змінних обставин і вимог, незважаючи на чітко визначену структуру і правила.

Різниця між фреймворком та методологією полягає в підході: *фреймворк* – це набір структурованих рекомендацій та інструментів, які можна адаптувати до потреб проекту, але він не надає детального покрокового плану; а *методологія* – це більш жорсткий і формалізований набір принципів, які регламентують виконання проектів від початку до кінця, з конкретними кроками та вимогами. Простіше кажучи, методологія каже «що і як треба робити», а фреймворк каже «є основа – адаптуй під себе».

Отже, методології можна, умовно, розділити на дві групи: традиційні (англ. predictive) та адаптивні (або гнучкі англ. adaptive / agile). Фактична поява гнучких методологій і створила поняття фреймворку.

Розглянемо кілька сучасних традиційних методологій:

Каскад (англ. Waterfall) – це послідовний підхід, у якому кожний етап проєкту виконується один за одним без повернення назад. Усі вихідні вимоги

визначаються до початку і зміни вже не вносять (або вкрай рідко). Перевагою є чітко зрозумілі цілі, а недоліком низька гнучкість. Часто використовується у будівництві, виробництві та оборонних проєктах.

PRINCE2 (англ. Projects IN Controlled Environments) – це традиційна методологія, орієнтована на контроль. Проєкти існують в контрольованому середовищі. Весь проєкт розбивають на керовані фази з контрольними точками для перевірки прогресу. Передбачає детальну документацію і формальний підхід до ухвалення рішень. Притаманна фінансовим сферам та державному секторі.

Метод «критичного шляху», CPM (англ. Critical Path Method) – це традиційна методологія, орієнтована на послідовність виконання задач, що і визначає загальну тривалість проєкту. Такий підхід допомагає виявляти завдання, що можуть виконуватись паралельно, оптимізувати графік. Гарно підходить для великих проєктів або портфелів зі складною структурою задач. Часто використовується у будівництві, виробництві та інженерії.

Управління проектами критичного ланцюга, CCPM (англ. Critical Chain Project Management) – це традиційна методологія, що є наступником CPM, яка враховує обмеження ресурсів. Орієнтована на реалістичне планування з ощадливим підходом до залучення команди без перевантаження і вигоряння. Зосереджена на управлінні залежностей та захисті критичного ланцюга (задач) від зривів. Передбачає використання «буфера часу», що покликані компенсувати непередбачені затримки. Підходить для проєктів із обмеженими ресурсами.

Six Sigma – це традиційна методологія, орієнтована на підвищення якості та зменшення дефектів. За основу береться статистичний аналіз процесів і даних. Передбачає дві моделі DMAIC (для покращення існуючих процесів) і DMADV (для створення нових). Високо дисциплінована методологія та вимагає фахівців із відповідним досвідом. Часто використовується у виробництві, логістиці та сервісах, де критична якість.

Також розглянемо кілька сучасних гнучких методологій:

Lean (Худий) – це гнучка методологія, орієнтована на цінність для клієнта при мінімальних необхідних витратах ресурсів, а особливо часу. Головні принципи: прибрати все зайве, що не додає цінності (muda), безперервне вдосконалення (kaizen). Фактично, схудни аби бути гнучким. Свій початок бере з 1950-х років, із впровадження її на виробництві Toyota двома засновниками – Тайіті Оно та Шігео Шінго. Спочатку застосовувалася в автомобільній промисловості, пізніше адаптована для проєктного менеджменту та IT. Lean не диктує жорстких ролей чи інструментів, а дає набір принципів, які можна реалізувати різними способами (Kanban, Scrum, тощо).

Agile (Гнучкий, спритний) – це гнучка методологія, орієнтована на швидку адаптацію, ітеративність та співпрацю. Agile є парасолькою над популярними фреймворками, тому його вважають «методологічним маніфестом» до управління проєктами.

Його головна ідея полягає в створенні робочого продукту малими частинами й циклічно, постійно отримуючи зворотний зв'язок від замовника. Офіційно Agile з'явився у 2001 році після публікації «Agile Manifesto» (Agile-маніфест) групою з 17 провідних розробників у Сноуберді, США. Він містить основні принципи та цінності, на яких базується гнучка методологія розробки програмного забезпечення (Agile). Маніфест зосереджений на ефективній співпраці, адаптивності та створенні продуктів, що мають реальну цінність для клієнтів.

Цінності Agile:

1. **Люди і взаємодія важливіші за процеси та інструменти**

   Фокус на ефективній комунікації та командній роботі, а не лише на дотриманні процесів.

2. **Працююче програмне забезпечення важливіше за вичерпну документацію**

   Краще мати працюючий продукт, ніж витрачати час на створення великої кількості документів.

3. **Співпраця з замовником важливіша за обговорення умов контракту**

   Постійна комунікація та зворотний зв'язок від замовника мають більшу цінність, ніж лише слідування контрактним вимогам.

4. **Готовність до змін важливіша за дотримання початкового плану**

   Процес розробки повинен бути гнучким, щоб мати змогу адаптуватися до змін і нових вимог.

Принципи Agile:

1. Основною метою є задоволення клієнта шляхом швидкої і безперервної поставки цінного програмного забезпечення.
2. Ласкаво просимо зміни вимог, навіть на пізніх стадіях розробки.
3. Постачання робочого програмного забезпечення відбувається через короткі проміжки часу (кілька тижнів – кілька місяців).
4. Замовники та розробники повинні працювати разом щодня протягом усього проекту.
5. Створюйте проекти навколо мотивованих людей і надайте їм необхідну підтримку та довіру.
6. Найкращий спосіб передачі інформації між командою – це особиста комунікація.
7. Працююче програмне забезпечення є основним показником прогресу.
8. Agile методології сприяють стійкому темпу роботи команди.
9. Постійна увага до технічної досконалості та гарного дизайну покращує гнучкість.
10. Простота – мистецтво максимально скорочувати обсяг зайвої роботи – є ключовим фактором.

11. Найкращі архітектури, вимоги та дизайни виходять з саморганізованих команд.
12. Регулярно команда аналізує свою роботу та коригує процеси для покращення ефективності.

Agile включає різні фреймворки: Scrum, Kanban, Extreme Programming (XP), SAFe, LeSS та інші.

Узагальнюючи, класифікацію методологій та фреймворків можна зобразити у вигляді схеми (див. рис. 2.6).

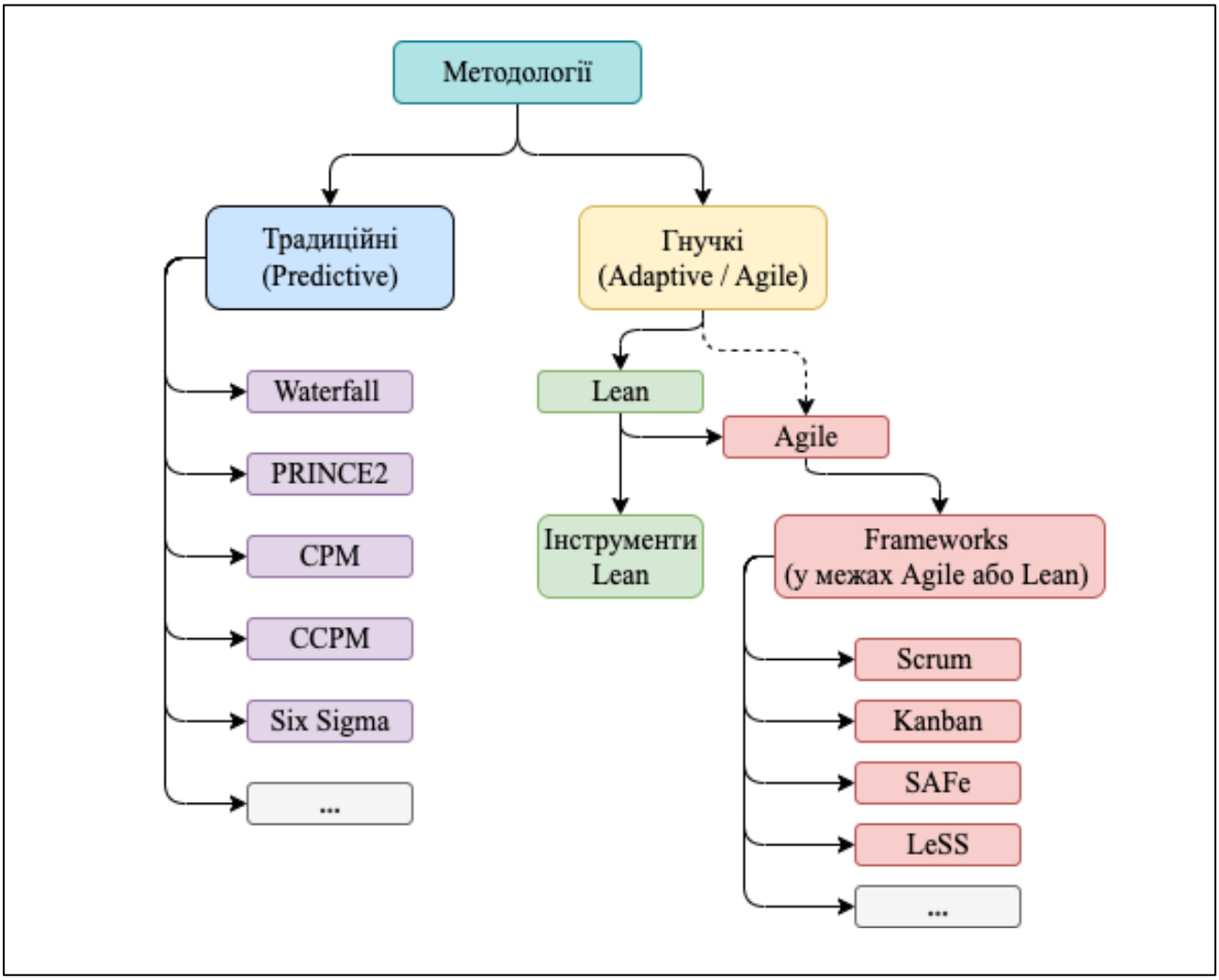

Рисунок 2.6 – Класифікація методологій та фреймворків.

Схема, яка наочно демонструє взаємозв'язки між Lean, Agile та Scrum наведена на рис. 2.7. На ній видно, що Lean охоплює Agile, а Agile, у свою чергу охоплює Scrum. Це добре ілюструє принцип, що Lean є більш широка філософія/підхід, тоді як Agile і Scrum – це її практичніші реалізації у рамках управління проєктами. При цьому Kanban, як фреймворк є частиною Agile, перетинається із фреймворком Scrum, і одночасно, є інструментом Lean.

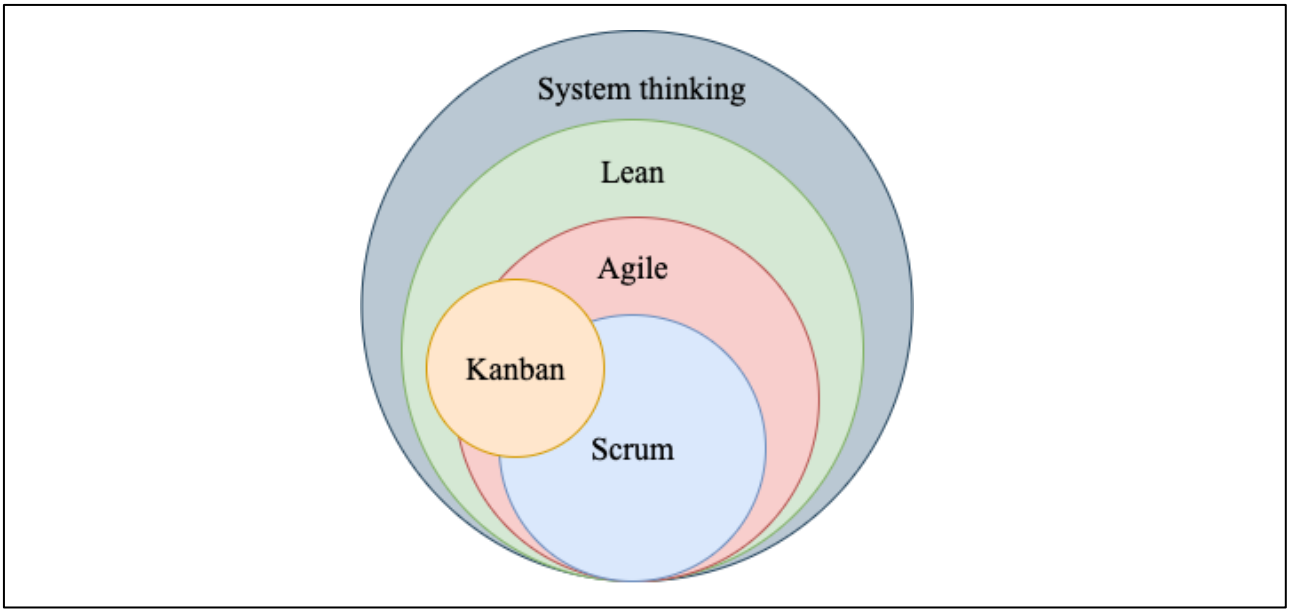

Рисунок 2.7 – Взаємозв'язок Lean, Agile, Scrum та Kanban.

Розглянемо кілька сучасних Agile-фреймворків:

Scrum (штовханина, сутичка у регбі) – найпопулярніший фреймворк Agile, що широко використовується в розробці програмного забезпечення, створений у 1990-х Кеном Швабером та Джеффом Сазерлендом. Схему життєвого циклу проєкту на основі Scrum показано на рис. 2.8.

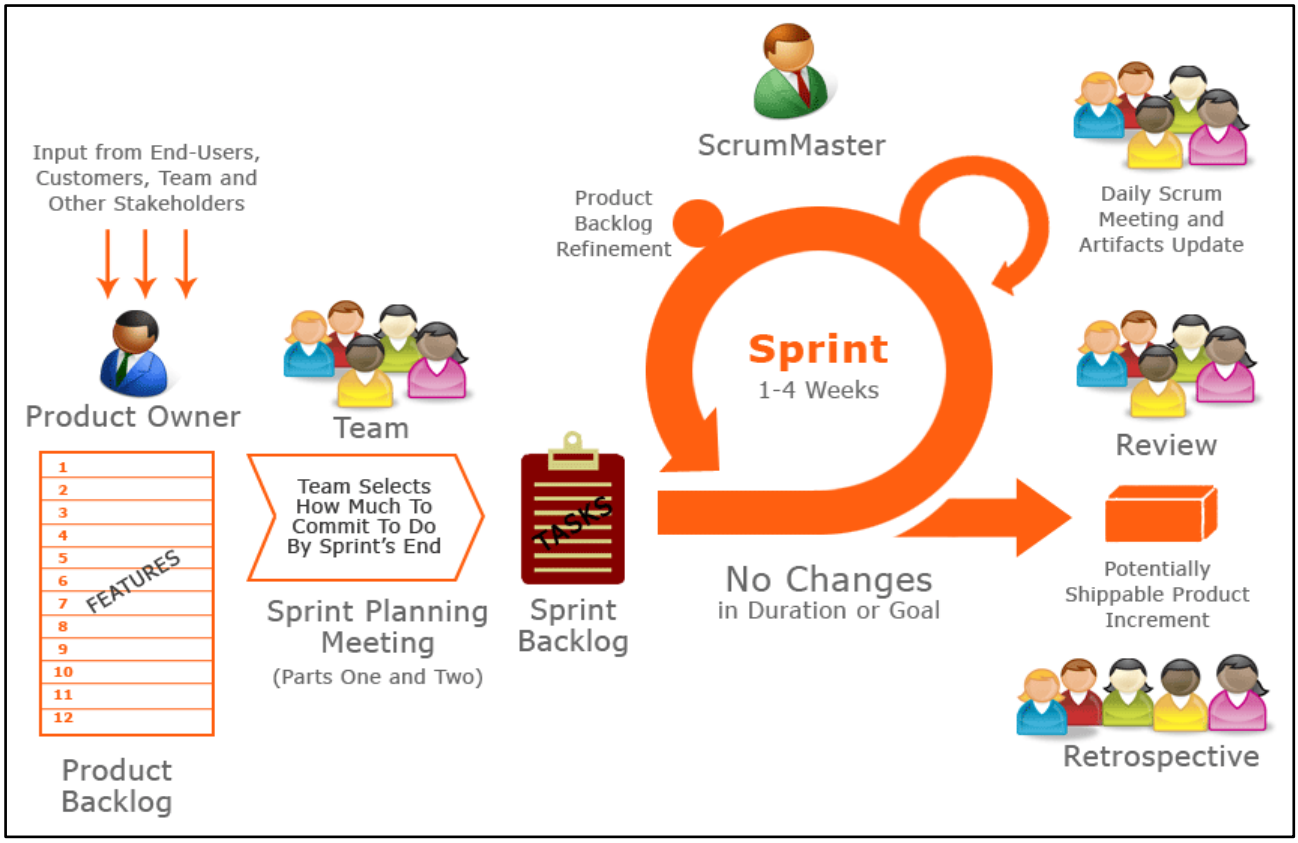

Рисунок 2.8 – Схема життєвого циклу проєкту на основі Scrum.

Він передбачає роботу команди у коротких ітераціях – спринтах, за результатом яких замовник отримує нову версію продукту, який називають інкрементом.

Ці спринти (цикли) тривають 1–4 тижні, а ролі (Scrum Master, Product Owner, Development Team), в команді проєкту та церемонії (щоденні стендапи, планування, ретроспектива), чітко визначені. Головна ідея: забезпечити швидку доставку цінності, розбиваючи роботу на керовані частини.

Як бачимо з рис. 2.8, вхідні вимоги надходять до власника продукту (Product Owner) від кінцевих користувачів, замовника та інших зацікавлених сторін (stakeholders), в результаті тісної комунікації між усіма учасниками процесу. Власник продукту формує продуктовий беклог (Product backlog – перелік доробок, функцій та бізнес вимог), з яким вже працює команда. Команда із цих вимог формує список тих доробок, що будуть реалізовані в наступному спринті – Sprint Backlog. На кожен спринт команда створює новий список задач, які будуть реалізовані. А протягом всього спринту виконують поставлені задачі. При цьому відбуваються щоденні зустрічі команди для обговорення поточних задач, проблем, статусів та прогресу. Вся команда знає, що саме відбувається із продуктом і хто якою задачею зайнятий.

За результатами спринта замовник отримує новий продукт із новими функціями. Всі невідповідності чи помилки повертаються в беклог наступного спринта новими задачами.

Scrum швидко адаптується до змін, оскільки ітераційний підхід спринтів дозволяє швидко реагувати на зміни в вимогах. Він гарантує прозорість, що забезпечується регулярними зустрічами, на яких всі учасники відслідковують прогрес. Scrum підтримує високий рівень залученості та відповідальності команди, хоч і вимагає зрілої команди, самостійної та дисциплінованої команди. При всіх перевагах, такий підхід важко масштабується на великі проєкти і організації.

Kanban (з японської «вивіска») – візуальний Agile-фреймворк управління проєктами, заснований на візуалізації робочого процесу та принципі безперервного вдосконалення. Він передбачає використання дошки з картками, які представляють завдання, і колонками, що відображають різні стадії їх виконання. Схему життєвого циклу на основі Kanban показано на рис. 2.9.

Kanban походить із Lean і Toyota Production System, але в 2000-х роках був адаптований до IT Девідом Андерсоном. Головні принципи: обмеження кількості завдань у роботі (Work In Progress limits, WIP limits), прозорість, безперервне вдосконалення процесу. Він дає змогу швидко побачити «вузькі місця» в процесі реалізації проєкту та реагувати на проблеми. Може використовуватися як самостійно, так і разом зі Scrum (Scrumban).

Kanban дошка мінімально має три колонки (рис. 2.9), оскільки мінімально в трьох статусах можна оцінювати задачу: «запит», «в роботі» і «виконано». Обмеження на кількість задач в роботі або певному статусі (WIP limits) сприяє

захисту команди від вигоряння і перевантаження. Також пріоритет задач визначається окремою поміткою (тегом), і означає, що задачі із вищим пріоритетом мають бути виконанні в першу чергу.

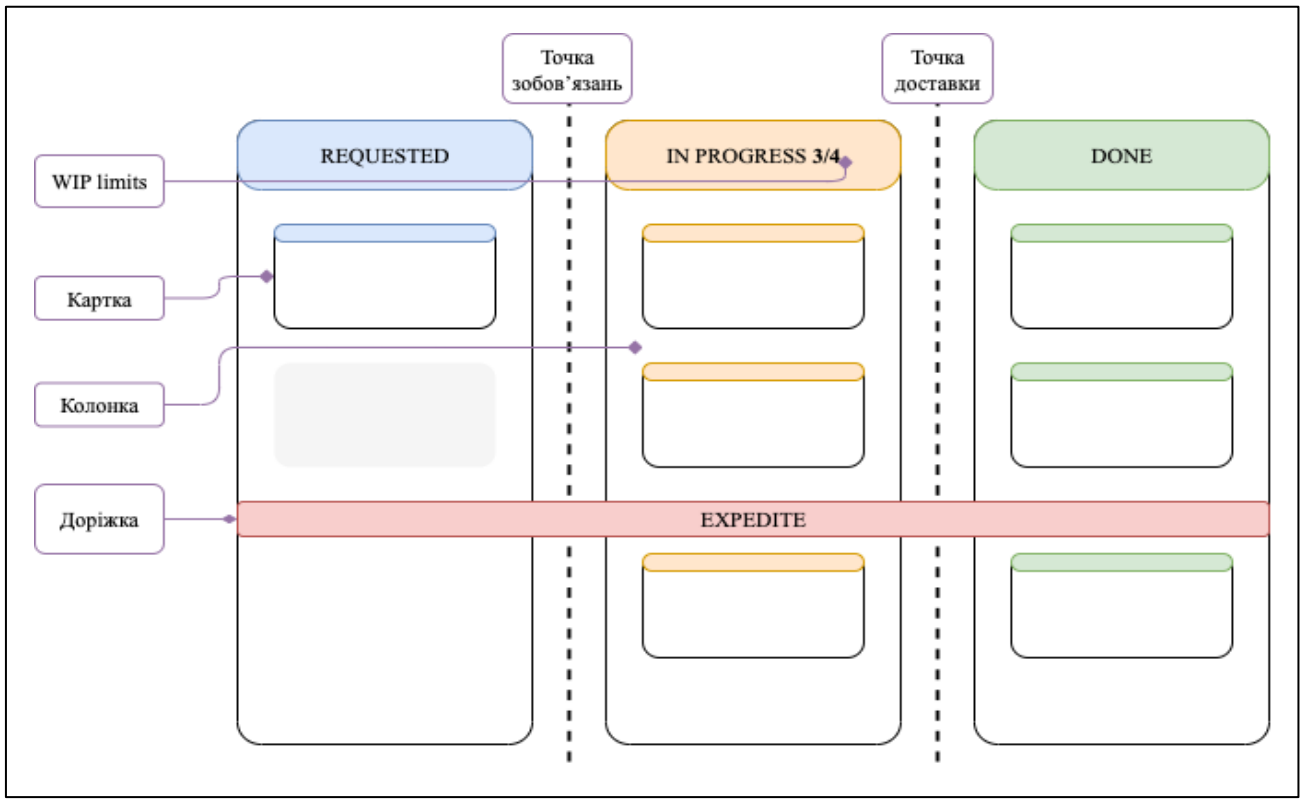

Рисунок 2.9 – Схема життєвого циклу проєкту на основі Kanban дошки.

Загалом, принципи візуалізації у вигляді дошки часто застосовують не тільки для ведення проектів, а й для обслуговування клієнтів, виробництва, послуг, тощо. Нижче наведено візуалізацію формування пріоритизацію задач в беклозі із подальшим їх винесенням в спринт (рис. 2.10).

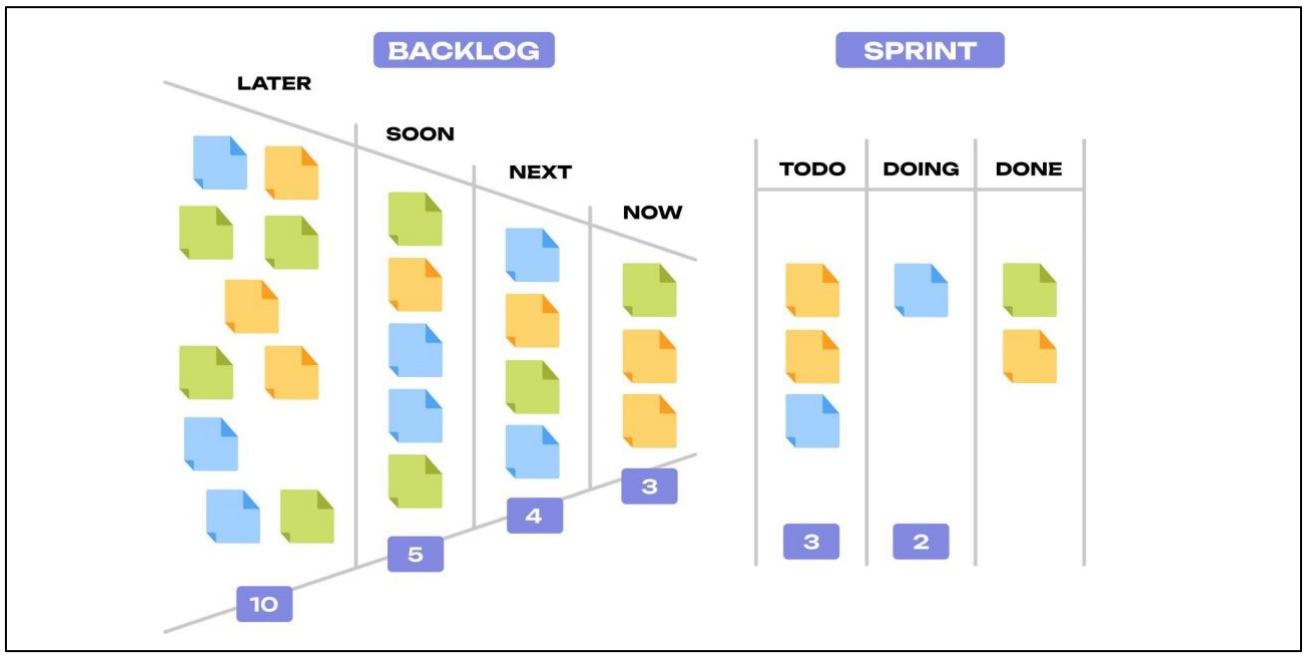

Рисунок 2.10 – Схема формування спринту на основі Kanban дошки.

Перевагами такого фреймворку є прозорість, візуалізація, простота гнучкість і головне зниження навантаження на команду. Kanban не вимагає ітерацій або жорстких термінів, що робить його адаптивним до різних умов.

Проте, відсутність структури, у порівнянні зі Scrum, може бути проблемою для великих команд. А відсутність конкретних ітерацій може призводити до розмивання цілей проєкту.

Масштабований Agile фреймворк, SAFe (англ. Scaled Agile Framework) – фреймворк, що передбачений для масштабування Agile на великі організації та програми, з урахуванням особливостей масштабних проєктів, р озроблений Діном Леффінгвеллом у 2011 році, активно розвивається PMI та Scaled Agile Inc.

SAFe об'єднує принципи Agile, Lean і DevOps, щоб синхронізувати роботу багатьох команд. Структурно складається з кількох рівнів: команда, програма, портфель, рішення (англ. Solution). Призначений для застосування там, де є потреба узгодити роботу сотень людей над одним продуктом чи системою.

Масштабований Scrum, LeSS (англ. Large-Scale Scrum) – фреймворк масштабування Scrum для кількох команд, що працюють над одним продуктом. Розроблений Крейгом Ларманом і Басом Водде, уперше описаний у книзі Large-Scale Scrum: More with LeSS (2010).

Зберігає більшість принципів класичного Scrum, але вводить додаткові правила координації. Мінімізує бюрократію, зберігаючи простоту та прозорість процесів. А головне, LeSS ефективний там, де потрібно масштабувати Scrum без втрати його гнучкості.

Повертаючись до традиційних методологій, що передбачають лінійний (або каскадний) підхід до виконання проєктів, варто відзначити, що в їх основі лежить *Діаграма Ганта* (рис 2.11). Вона якнайкраще ілюструє послідовність всіх етапів і задач проєкту, його критичний шлях, зв'язок між ресурсами, тощо.

Засновником проєктного управління вважають американського інженера-механіка Генрі Ґанта, якого називають батьком технік планування та контролю. Саме він в 1910 році розробив перший формат своєї діаграми (графіка).

Діаграма Ґанта являє собою стовпчикову діаграму: відрізки, розміщені на горизонтальній шкалі часу, позначають початок та кінець виконання задачі. Кожен відрізок відповідає окремому завданню або підзадачі, на діаграмі легко зобразити вкладеність або декомпозувати задачу, розклавши її на менші. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. Також задачі на діаграмі показуються із врахуванням залежність між ними.

В лівій частині розміщується таблиця зі списком робіт, з урахуванням їх ієрархії. Кожен рядок відповідає задачі на часовій шкалі діаграми Ґанта.

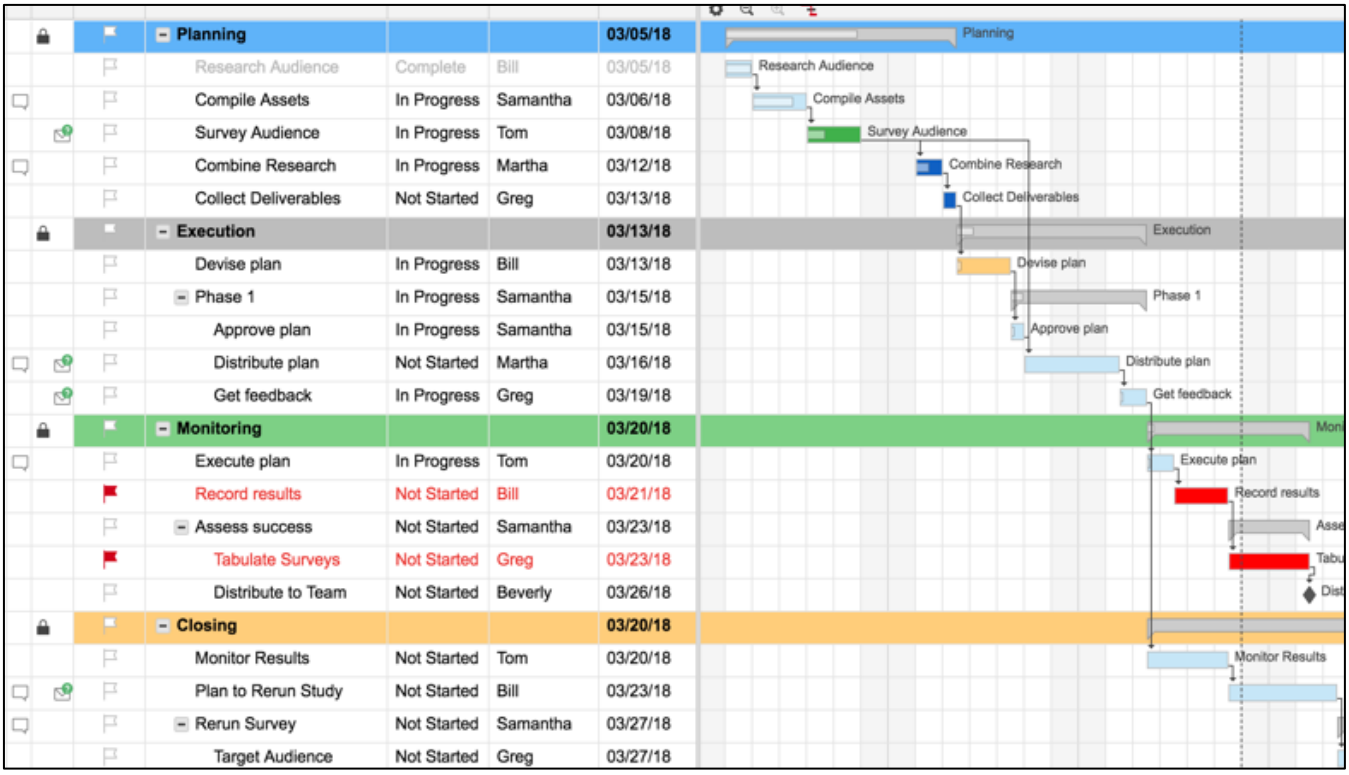

Рисунок 2.11 – Приклад діаграми Ґанта.

Основні елементи діаграми Ґанта:

- **Завдання (англ. Tasks)**: Кожне завдання проекту представлено як горизонтальна смуга на діаграмі. Довжина смуги відповідає тривалості завдання.
- **Шкала часу (англ. Timeline)**: Вона розташована вздовж верхньої або нижньої частини діаграми і може бути розбита на дні, тижні, місяці або інші часові інтервали залежно від тривалості проекту.
- **Взаємозв'язки між завданнями (англ. Dependencies)**: Відображають, які завдання повинні бути завершені до початку інших. На діаграмі це зазвичай показано стрілками, що з'єднують кінці однієї смуги з початком іншої.
- **Мітки етапів (англ. Milestones)**: Важливі точки проекту, такі як завершення фази або досягнення певної мети, можуть бути позначені як окремі події.
- **Відсоток виконання (англ. Progress)**: На смугах завдань може відображатися прогрес виконання у вигляді заповнених частин смуги.

Вона використовується в традиційних методах управління проектами, де є чітка послідовність етапів і важлива структура виконання завдань у визначений час. Цей інструмент допомагає візуально планувати і контролювати виконання завдань у проектах, де акцент робиться на дотриманні графіка і фіксованих строках.

Хоча діаграму Ганта також використовують в різних методологіях і гнучких фреймворках, оскільки це дуже ефективний інструмент планування та оцінки прогресу.

Стандарти, настанови, методології та фреймворки проєктного менеджменту забезпечують теоретичну основу і найкращі практики, які можна адаптувати до будь-якого проєкту, незалежно від його типу або галузі.

Багато сучасних фреймворків надають конкретні інструменти та техніки для управління проєктами в різних умовах. Вибір між методології та фреймворка, або їх комбіноване використання, залежить від специфіки проекту, команди та організації.

2.7 Програмні комплекси для управління проєктами

У управлінні проєктами важливо використовувати ефективні програмні комплекси, які допомагають організувати, контролювати і управляти всіма аспектами проекту.

Найпопулярніший та найвідоміший з інструментів для управління проектами це *Microsoft Project*. Ця програма забезпечує розробку планів, розподіл ресурсів, відстеження прогресу та аналіз обсягів робіт, а також дозволяє створювати розклади з урахуванням критичного шляху. Візуалізація здійснюється у вигляді діаграми Ґанта.

MS Project один з найдавніших програмних комплексів, що дозволяє створювати графіки проєктів, розподіляти ресурси, відстежувати прогрес і витрати. Це дуже потужний інструмент для створення звітів і візуалізацій даних, і він легко інтегрується з іншими продуктами Microsoft.

Проте, його вартість для малого бізнесу зависока, і специфіка роботи вимагає тривалого навчання та підготовки, до того ж, не спеціалізується на Agile чи інших гнучких методологій.

Atlassian JIRA – популярний інструмент для управління проєктами, особливо в середовищі розробки програмного забезпечення. Розроблений компанією Atlassian, Jira підтримує Scrum і Kanban фреймворки.

Ця система спеціалізується на відстеженні помилок в ході виконання проєкту, призначена для організації спілкування з командою, і для управління проєктами. Розроблена компанією Atlassian у 2002 році. Назва системи (JIRA) отримано шляхом модифікації японської назви Ґодзіла («Gojira»), що своєю чергою є алюзією на назву конкурентного продукту – Bugzilla.

JIRA підтримує Scrum, Kanban та інші Agile-фреймворки, легко інтегрується з системами контролю версій та дозволяє не просто слідувати процесам, а налаштовувати їх під потреби проєкту (створювати свої поля, робочі процеси, тощо).

Asana, Inc. – це веб- і мобільна платформа для «керування роботою». Asana, Inc. заснована в 2008 році Дастіном Московіцем (співзасновник Facebook і віце-президент з інженерних питань) і Джастіном Розенштейном. Вони познайомилися у Facebook, і створили інструмент продуктивності під назвою Tasks. А у 2008 році покинули Facebook, щоб створити Asana і запустили безкоштовну бета-версію в листопаді 2011 року. В квітні 2012 року Asana вийшла в комерційну експлуатацію.

Asana має інтуїтивно зрозумілий інтерфейс і просте налаштування, дозволяє створювати діаграми Ґанта, трекери завдань і Kanban-дошки і також дозволяє комунікувати з командою.

Trello – це безкоштовна система управління завданнями та проектами на основі канбан-дошок, розроблена у 2011 році компанією Fog Creek Software (тепер Glitch) і продана Atlassian в січні 2017 року. Вона заснована на Kanban фреймворку, а проєкти зображуються дошками, що містять списки, а списки містять картки.

Trello дуже зручний інструмент, що має безкоштовну ліцензію у базовій версії. Проте, його функціональність обмежена для складних проектів, відсутні розширені функції звітності.

Вибір програмного комплексу для управління проєктами є дуже важливим оскільки покликаний закрити потреби всіх учасників проєкту в комунікації та системній організації. До вибору варто підходити уважно, адже від специфіки проєкту, потреб команди та бюджету залежить який програмний комплекс підійде найкраще.

Microsoft Project підходить для великих і складних проектів, Jira ідеальна для команд, що використовують Agile методології, Asana і Trello є відмінними для менш складних проектів і простих завдань.

3. БІЗНЕС-АНАЛІЗ

3.1 Основи бізнес-аналізу

Запорукою успішного проєкту є якісно сформовані вимоги та потреби всіх зацікавлених осіб, учасників та клієнтів. Для цього варто проводити збір вимог та аналіз потреб всіх дотичних до проєкту (чи майбутнього продукту) осіб та організацій. Ця стадія називається бізнес-аналіз.

Бізнес-аналіз – це процес дослідження потреб організації або проекту з метою розробки ефективних рішень. Він фокусується на виявленні проблем і можливостей, формуванні вимог, оптимізації бізнес-процесів та забезпеченні інтеграції нових рішень у бізнес-структуру. Це практика створення умов для змін в організації (проєкті), шляхом визначення потреб та рекомендацій рішень, які приносять користь зацікавленим сторонам.

Бізнес-аналіз передбачає збирання, дослідження та обробку даних про організацію, її ринок, конкурентів, процеси і потреби. Основною метою є пошук оптимальних шляхів для досягнення поставлених цілей, покращення продуктивності або адаптації до змін.

Ключовою фігурою є бізнес-аналітик. Його включають в проєктну команду саме для ефективної комунікації між усіма учасниками та зацікавленими сторонами. Він є посередником між бізнесом і технічними командами. Його завдання зрозуміти потреби бізнесу та перетворити їх на конкретні вимоги для розробників. Фактично, грає роль перекладача, але при цьому він формулює вимоги бізнесу. Аналітик допомагає організації визначити, які процеси варто автоматизувати або змінити, які рішення допоможуть оптимізувати роботу компанії. Ключовими завданнями бізнес-аналізу є:

- Формулювання потреб бізнесу;
- Обґрунтування необхідності впровадження змін;
- Опис рішення (продукту), що задовольняє вимоги бізнесу.

У кожному проекті існують зацікавлені сторони (stakeholders). Це можуть бути власники бізнесу, менеджери, співробітники, користувачі та інші групи. Кожна з цих груп має свої інтереси та очікування, і бізнес-аналітик повинен врахувати їх при формуванні вимог. Ключові завдання бізнес-аналітика:

- Виявлення потреб зацікавлених сторін (клієнтів, співробітників);
- Формування вимог для проекту;
- Оцінка впровадження рішень у бізнес-процеси;
- Комунікація між технічною командою та керівництвом.

У проєкті велику роль відіграють зацікавлені сторони (*stakeholders*). Це можуть бути власники бізнесу, менеджери різних ланок, співробітники, користувачі продукту та інші групи. Всі вони мають свої інтереси та очікування від результату або майбутнього продукту чи послуги, яку покликаний створити

проєкт, і бізнес-аналітик повинен врахувати їх всі при формуванні опису майбутнього рішення, що задовольнить всі їх потреби.

Бізнес-аналіз є невід'ємною частиною процесу розробки ефективних бізнес-рішень. Він необхідний, коли бізнес чітко розуміє, що має певні проблеми, проте не має чітких та ефективних шляхів до їх вирішення, або має розуміння, проте не володіє технічними навичками (наприклад, IT розробки).

Бізнес-аналітик виступає мостом між бізнесом і технічними фахівцями, виконавцями, забезпечуючи, щоб вимоги були чітко сформовані і вирішували реальні потреби організації та зрозумілими і конкретними.

Всі вимоги, які збирає бізнес-аналітик можуть бути бізнес-вимогами, користувацькими, функціональними та нефункціональними вимогами.

Бізнес-вимоги: це чіткий опис того, чого хоче досягти бізнес за допомогою проєкту, при цьому нехтують відповіддю на питання «як це буде зроблено?». В межах одного проєкту можливе або вирішення бізнес-проблеми, або бізнес-можливості. Саме такий підхід є запорукою успіху проєкту, адже поєднуючи вирішення проблем із реалізацією можливостей, виникає ризик провалу проєкту. Бізнес-вимоги це високорівневі цілі та потреби, які організація намагається задовольнити.

Як результат збору бізнес вимог формується офіційний документ – *Документ бізнес-вимог (англ. Business Requirements Document, BRD)*. Він визначає бізнес-цілі, обсяг і вимоги високого рівня проєкту. Цей документ служить інструментом комунікації, який усуває розрив між зацікавленими сторонами та командою проєкту, задля узгодженості дій в прагненні реалізувати очікування проєкту. Прикладом бізнес-вимог можуть бути: «зниження витрат», «збільшення прибутків» чи «оптимізація процесів», іншими словами це «хотілки» бізнесу.

Користувацькі вимоги: це опис того, чого хочуть досягти зацікавлених сторін, які описують, як користувачі будуть використовувати систему або продукт. Вони визначають завдання та цілі, які користувачі зможуть досягти за допомогою системи, а не конкретні технічні деталі реалізації.

Функціональні вимоги: це детальний опис того, що програмне забезпечення повинно робити для задоволення потреб користувачів та досягнення бізнес-цілей. Вони визначають конкретні функції, операції та поведінку системи.

Нефункціональні вимоги: це детальний опис атрибутів якості системи, які впливають на її роботу, зручність використання та загальну задоволеність користувачів. На відміну від функціональних вимог, які описують, що система повинна робити, нефункціональні вимоги описують, як система повинна це робити (вимоги до безпеки, продуктивності, тощо).

Процес бізнес-аналізу, починається із збору потреб, хоча складається із п'яти послідовних етапів (див. рис. 3.1).

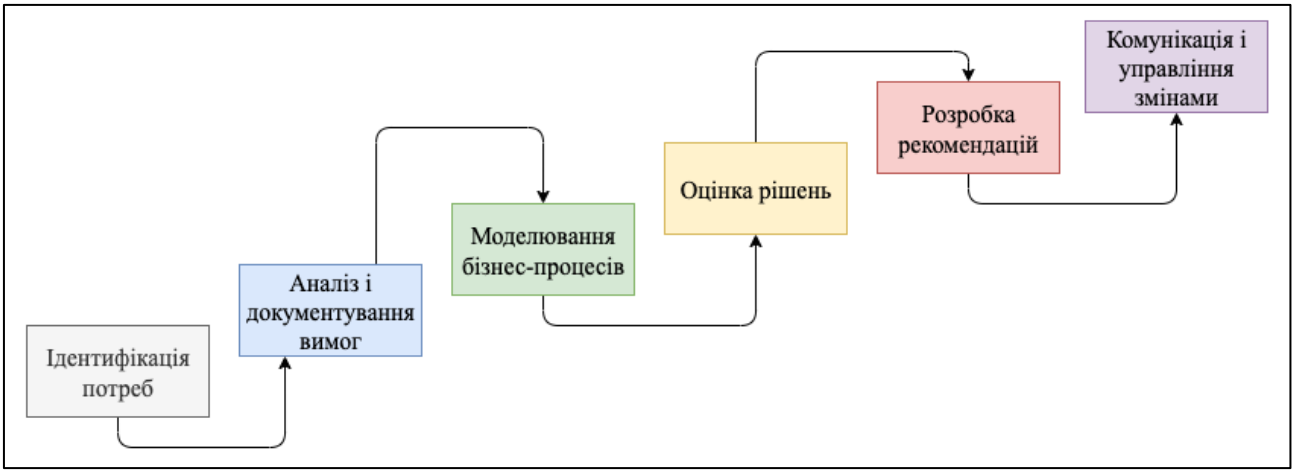

Рисунок 3.1 – Процес бізнес-аналізу.

Ідентифікація потреб: перший етап процесу бізнес-аналізу, на якому варто зібрати інформацію від усіх зацікавлених осіб, включаючи клієнтів, користувачів, менеджерів та інших осіб, що впливатимуть на результат. Збір вимог можна проводити і різний спосіб це проведення інтерв'ю, опитуваня, створення і робота із фокус-групами, аналіз існуючих процесів і систем для виявлення потреб і проблем.

Як показує практика, найкращий спосіб збору вимог це дати можливість зацікавленим особам заповнити опитувальник (бріф). Такий підхід уніфікує всі вимоги, створює систему для легкої і зручної подальшої обробки, а також ніяк не обмежує тих, хто заповнює цей документ.

Аналіз і документування вимог: на цьому етапі бізнес-аналізу відбувається визначення і документування чітких і зрозумілих вимог до проекту. Тобто, бізнес-аналітик створює документацію, що необхідна для опису всіх функціональних вимог такої як специфікації вимог, моделі процесів, діаграми випадків використання (use cases).

Моделювання бізнес-процесів: етап на якому, використовуючи різні техніки здійснюють моделювання для візуалізації бізнес-процесів і вимог. Застосування діаграм потоку даних (англ. Data Flow Diagram, DFD), нотація моделювання бізнес-процесів (англ. Business Process Model and Notation, BPMN), та інших інструментів для створення зрозумілих і детальних моделей.

Оцінка рішень: аналіз можливих рішень для визначення їх відповідності до вимог і потреб бізнесу. Проведення оцінки варіантів, розрахунок витрат і вигод, аналіз впливу на існуючі процеси.

Розробка рекомендацій: на основі аналізу і оцінки розробка рекомендацій щодо реалізації рішень. Підготовка рекомендацій по вдосконаленню процесів, вибору технологій, або впровадженню нових рішень.

Комунікація та управління змінами: комунікація з усіма зацікавленим сторонам для донесення (повідомлення) результатів аналізу. Управління змінами

у вимогах, забезпечення контролю над реалізацією і перевірка відповідності рішень встановленим вимогам.

3.2 Техніки бізнес-аналізу

Техніки та інструменти бізнес-аналізу допомагають бізнес-аналітикам систематизувати, аналізувати та документувати вимоги до проекту. Кожна техніка має свої переваги і використовується в певних ситуаціях. Детально розглянемо найбільш поширен техніки і інструменти бізнес-аналізу, а також правила їх використання.

SWOT-аналіз – це простий, але дуже потужний метод стратегічного планування, який дозволяє оцінити внутрішні та зовнішні фактори, що впливають на успішність організації чи проекту. SWOT – це абревіатура, яка розшифровується як:

S (Strengths) – Сильні сторони.

W (Weaknesses) – Слабкі сторони.

O (Opportunities) – Можливості.

T (Threats) – Загрози.

Сильні сторони (Strengths): це ті характеристики компанії або проекту, які надають йому переваги над конкурентами. Це можуть бути унікальні ресурси, кваліфікований персонал, інноваційні технології або добре відпрацьовані бізнес-процеси.

Слабкі сторони (Weaknesses): це фактори, які можуть заважати організації або проекту досягти успіху. Це можуть бути недостатня кваліфікація персоналу, погано організовані процеси, низький рівень інвестицій або слабка репутація на ринку.

Можливості (Opportunities): це зовнішні фактори, які можуть сприяти розвитку компанії або проекту. Наприклад, нові ринки, технологічні інновації, зростання попиту на продукцію.

Загрози (Threats): це зовнішні фактори, які можуть негативно вплинути на організацію або проект. Це можуть бути економічні кризи, зростання конкуренції, зміни у законодавстві або змінені споживацькі переваги.

Приклад SWOT-аналізу наведено на рис. 3.2.

Рисунок 3.2 – Приклад SWOT-аналізу.

Приклад SWOT-аналізу: уявімо компанію, яка виробляє екологічно чисті товари. Її сильними сторонами можуть бути інноваційні продукти та зростаючий попит на екологічну продукцію. Слабкі сторони – висока ціна товарів. Можливості – розширення на нові ринки, а загрози – конкуренція з великими виробниками, які можуть знизити ціни.

PESTLE-аналіз – це інструмент для оцінки зовнішніх факторів, які можуть впливати на організацію або проект. Він охоплює шість ключових аспектів зовнішнього середовища: політичні, економічні, соціальні, технологічні, правові та екологічні фактори. Розглянемо графічне представлення методу (рис. 3.3) у вигляді схеми.

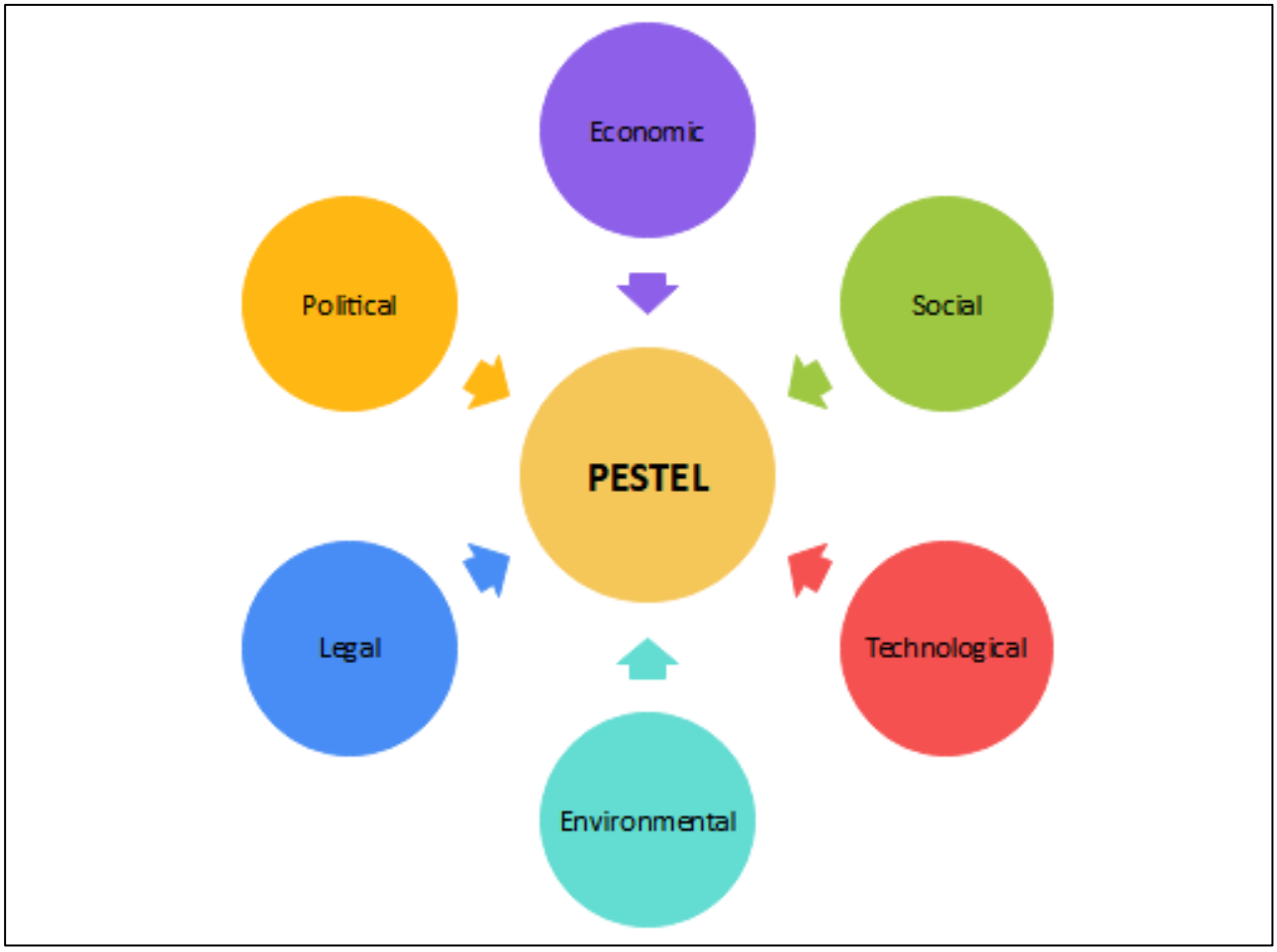

Рисунок 3.3 – Приклад PESTLE -аналізу.

Політичні фактори (Political): це державна політика, регулювання та зміни у владі, які можуть вплинути на бізнес. Наприклад, нові закони, державні програми підтримки або зміни в оподаткуванні.

Економічні фактори (Economic): це економічна ситуація може суттєво вплинути на бізнес. Наприклад, рівень інфляції, зміни валютних курсів, процентні ставки і загальний рівень економічного зростання.

Соціальні фактори (Social): це демографічні зміни, культурні уподобання, рівень освіти та здоров'я населення. Наприклад, зміна стилю життя людей може вплинути на попит на певні продукти.

Технологічні фактори (Technological): це розвиток нових технологій може як відкривати нові можливості для бізнесу, так і створювати нові загрози. Наприклад, автоматизація процесів або впровадження штучного інтелекту.

Правові фактори (Legal): це **з**акони та регулювання відіграють важливу роль у діяльності бізнесу. Це можуть бути зміни в трудовому законодавстві, вимоги до якості продукції або правила захисту споживачів.

Екологічні фактори (Environmental): це зміни в екологічних стандартах або кліматичні зміни можуть вплинути на бізнес, особливо в секторах, де важливе екологічне регулювання, таких як виробництво чи сільське господарство.

Модель 5 сил Портера (метод аналізу) – це стратегічний інструмент (рис. 3.4), який допомагає зрозуміти конкурентне середовище ринку. Вона аналізує п'ять основних сил, які впливають на здатність компанії досягти успіху на ринку.

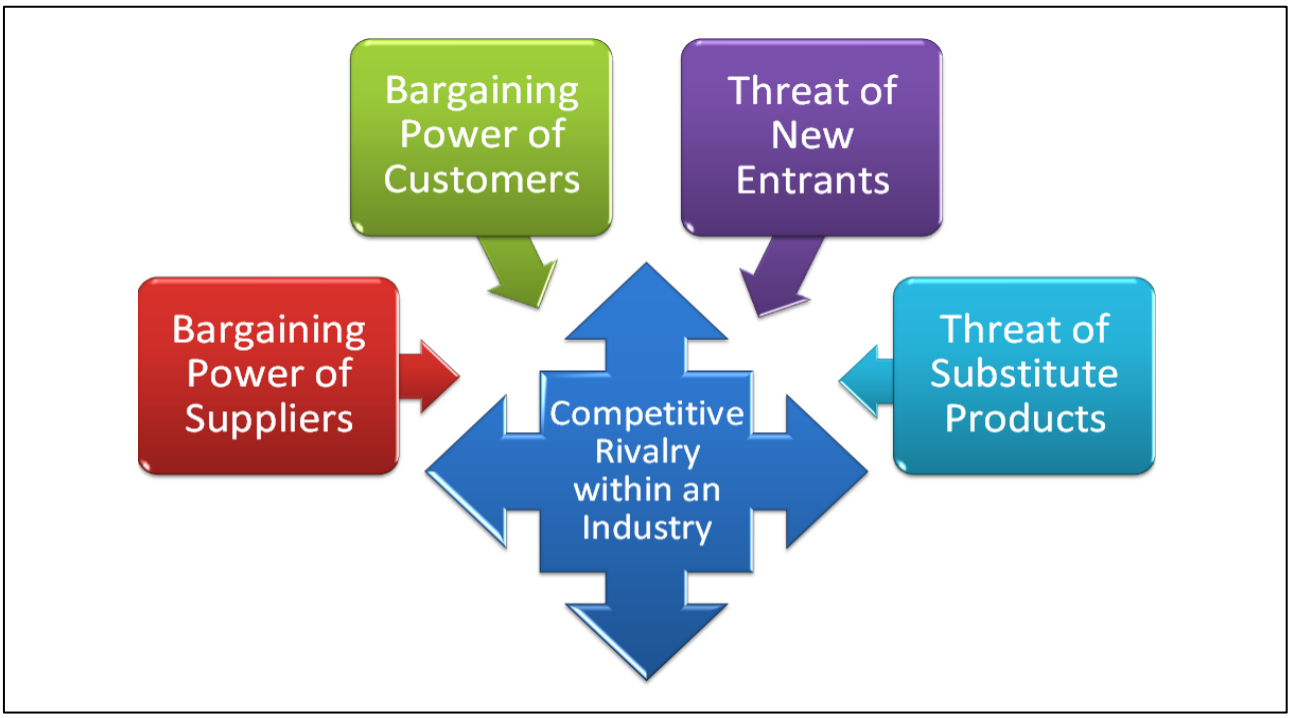

Рисунок 3.4 – Приклад аналізу 5 сил Портера.

Сила 1: Конкуренція всередині галузі

Це рівень конкуренції між існуючими гравцями на ринку. Чим більше компаній конкурують між собою, тим важче зайняти міцну позицію на ринку.

Сила 2: Загроза нових гравців

Нові компанії можуть увійти на ринок і знизити ринкову частку існуючих гравців. Чим більше бар'єрів для входу на ринок (наприклад, високі стартові витрати), тим менша загроза нових конкурентів.

Сила 3: Загроза замінників

Замінники – це альтернативні продукти або послуги, які можуть виконати ті ж функції, що й існуючий продукт. Наприклад, електронні книги є замінниками паперових.

Сила 4: Сила постачальників

Постачальники мають силу, якщо вони можуть впливати на ціни або якість своїх товарів або послуг. Якщо постачальників мало, їхня сила зростає, і це може вплинути на прибутковість компанії.

Сила 5: Сила покупців

Покупці мають більше сили, якщо можуть легко змінити постачальника або мають доступ до великої кількості альтернатив. Якщо клієнт може впливати на ціни або умови, це може негативно вплинути на прибуток компанії.

Аналіз конкурентів – це метод бізнес-аналізу, який дозволяє зрозуміти, як організація позиціонується на ринку відносно інших компаній. Цей метод полягає в дослідженні сильних і слабких сторін ваших конкурентів, їхньої стратегії та ринкової позиції. На практиці являє собою збір ключових показників у таблицю для подальшого аналізу. При цьому напір параметрів, що порівнюються визначаються для кожного окремого дослідження окремо. Цілі методу аналізу конкурентів**:**

1. Визначити прямих та непрямих конкурентів;
2. Оцінити ринкові стратегії конкурентів;
3. Порівняти продукти або послуги конкурентів з вашими;
4. Виявити конкурентні переваги і слабкості.

Процес аналізу складається із чотирьох етапів:

1. *Визначення основних конкурентів.*

   Це можуть бути як прямі конкуренти (ті, що продають аналогічний продукт), так і непрямі (ті, що пропонують альтернативні рішення).

2. *Оцінка конкурентних переваг.*

   Це можуть бути краща якість продукту, краща ціна або інноваційність. Потрібно зібрати інформацію про те, в чому конкуренти кращі, і в чому вони поступаються.

3. *Порівняння стратегії.*

   Аналізуйте, які маркетингові та операційні стратегії використовують конкуренти.

4. *Аналіз сильних та слабких сторін.*

   Створіть таблицю, де перераховані сильні та слабкі сторони кожного конкурента.

Аналіз конкурентів дозволяє краще зрозуміти, як організація може зміцнити свою ринкову позицію, виявити можливості для покращення і загрози з боку ринку.

Аналіз зацікавлених сторін (stakeholder analysis) – це процес визначення, оцінки та управління зацікавленими сторонами проекту або організації (рис. 3.5). Зацікавлені сторони – це всі особи або групи, які мають інтерес до результатів проекту і можуть вплинути на нього або бути під впливом його результатів.

До зацікавлених сторін можуть належати:

- Власники бізнесу або інвестори.
- Менеджери проекту.
- Співробітники компанії.
- Постачальники.

- Клієнти або кінцеві користувачі продукту.
- Державні установи чи регулятори.

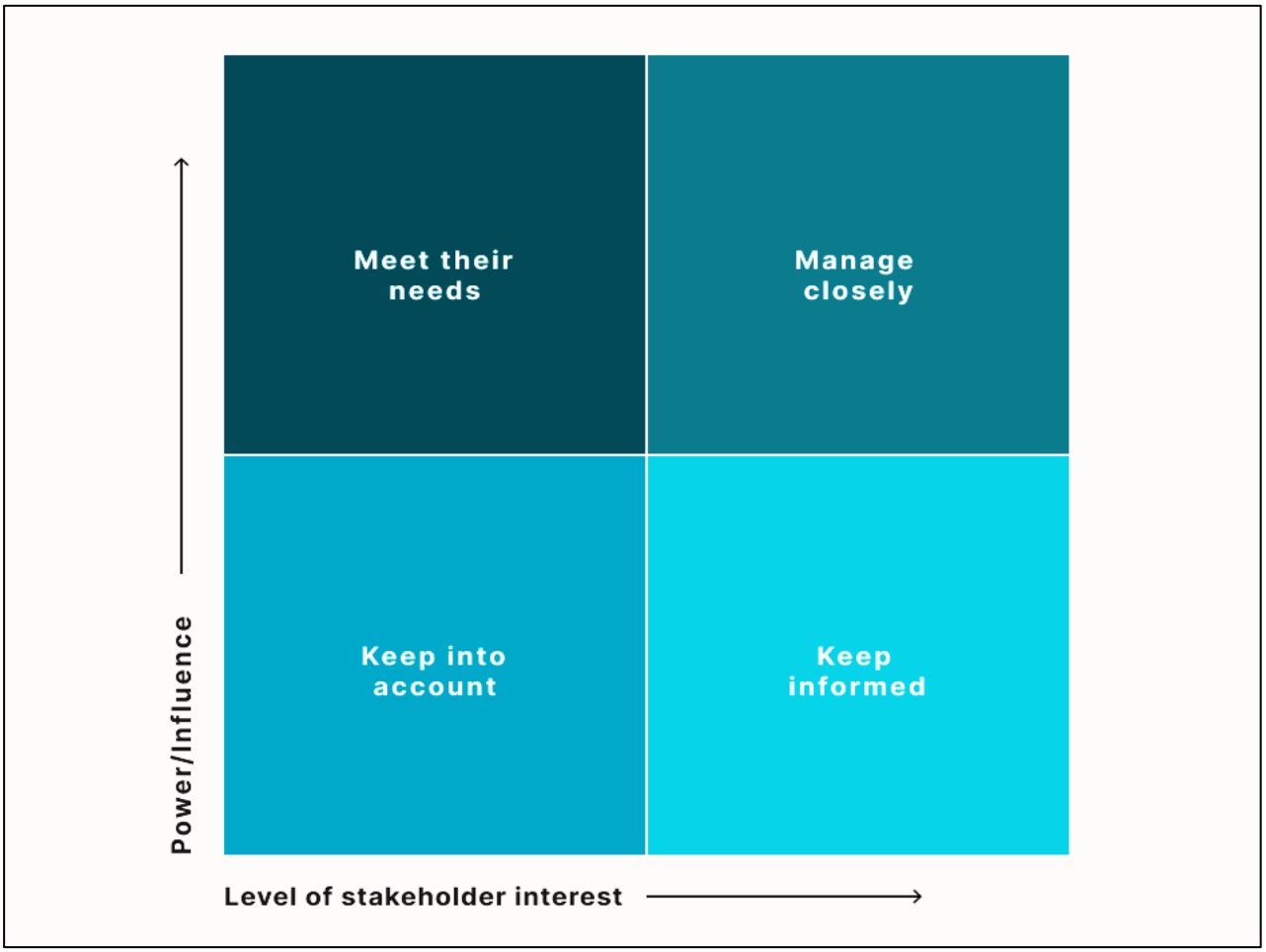

Рисунок 3.5 – Схема аналізу зацікавлених сторін.

Етапи аналізу зацікавлених сторін**:**

1. *Ідентифікація зацікавлених сторін*

   На першому етапі визначаються всі можливі особи та групи, які мають інтерес до проекту. Це включає не лише тих, хто прямо бере участь у проекті, але й непрямих учасників.

2. *Оцінка впливу та зацікавленості*

   Після ідентифікації важливо оцінити, наскільки кожен учасник може впливати на проект та який рівень їхньої зацікавленості.

3. Стратегія управління зацікавленими сторонами

   Для кожної групи або особи формується план взаємодії. Зацікавлені сторони з високим рівнем впливу і зацікавленості потребують більше уваги та комунікацій.

Щоб краще зрозуміти, як взаємодіяти із зацікавленими сторонами, використовується *матриця впливу та зацікавленості* (англ. Power/Interest Matrix). Вона має чотири квадранти:

- **Високий вплив, висока зацікавленість** – ключові гравці, з якими потрібно працювати максимально тісно.
- **Високий вплив, низька** зацікавленість – ті, кого потрібно інформувати і залучати у важливі моменти.
- **Низький вплив, висока зацікавленість** – зацікавлені, але не мають значного впливу. Їм варто надавати інформацію.
- **Низький вплив, низька зацікавленість** – варто лише тримати в курсі.

Аналіз зацікавлених сторін допомагає краще управляти проектом, забезпечуючи, що інтереси всіх сторін враховані, а ризики мінімізовані.

Аналіз ціннісного ланцюга (англ. Value Chain Analysis, VCA) – це модель (рис. 3.6), яка дозволяє визначити, які бізнес-операції створюють найбільшу додану вартість для продукту або послуги. Модель була розроблена Майклом Портером і допомагає зрозуміти, як різні етапи виробництва або надання послуг впливають на загальну конкурентоспроможність компанії.

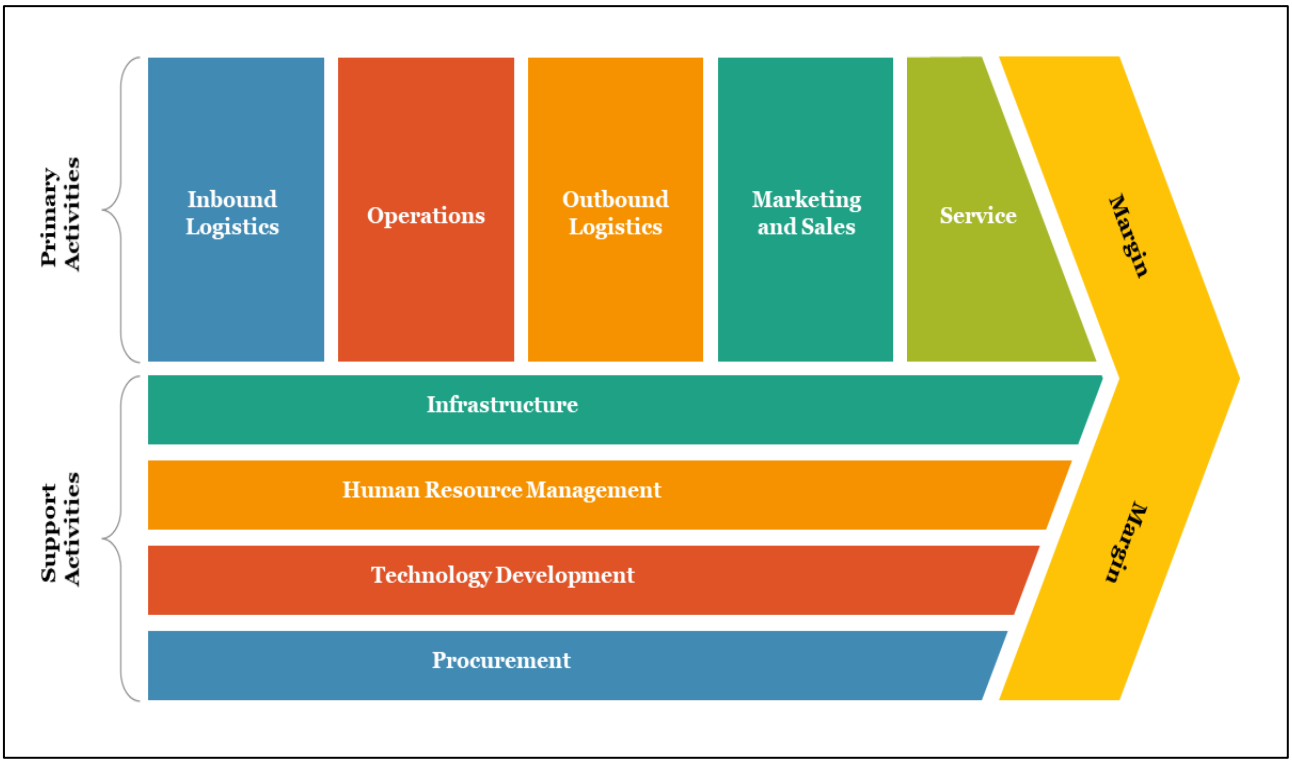

Рисунок 3.6 – Схема аналізу ціннісного ланцюга.

Ціннісний ланцюг складається з ряду операцій, які можна розділити на дві категорії:

Основні види діяльності (англ. Primary Activities) – ті, що безпосередньо пов'язані зі створенням продукту або послуги:

- *Логістика (вхідні постачання)* – отримання і зберігання сировини або матеріалів.
- *Операції* – процес виробництва або надання послуги.

- *Логістика (вихідні постачання)* – доставка готового продукту до клієнта.
- *Маркетинг та продажі* – просування і продаж товарів чи послуг.
- *Обслуговування* – післяпродажне обслуговування клієнтів.

Допоміжні види діяльності (англ. Support Activities) – ті, що підтримують основні процеси:

- *Інфраструктура компанії* – управління, фінанси, правова підтримка.
- *Управління людськими ресурсами* – підбір, навчання, розвиток персоналу.
- *Розробка технологій* – впровадження нових технологій у процеси.
- *Закупівлі* – придбання сировини, матеріалів або обладнання.

Кожна з цих діяльностей додає певну цінність до продукту або послуги. Основна мета аналізу ідентифікувати, на якому етапі можна підвищити ефективність або знизити витрати, щоб збільшити загальну цінність для клієнта.

Наприклад, компанія, що виробляє автомобілі, може покращити свій ціннісний ланцюг, автоматизувавши виробничі операції або покращивши логістику для зниження витрат на доставку.

Аналіз ціннісного ланцюга дозволяє компаніям зосередитися на тих аспектах, які забезпечують найбільшу додану вартість і підвищують їх конкурентоспроможність.

Моделювання бізнес-процесів – це техніка, яка дозволяє візуалізувати та описати, як функціонують процеси в організації. Мета цього процесу – ідентифікація неефективностей та покращення способів виконання завдань.

Бізнес-процес – це послідовність дій або операцій, які компанія виконує для досягнення конкретної мети. Це може бути процес виробництва продукції, обслуговування клієнтів або управління фінансами.

Існує багато стандартів та підходів до моделювання бізнес-процесів. Найпопулярніші з них:

- **BPMN (Business Process Model and Notation)** – це стандартизована мова для опису процесів. Вона дозволяє легко зрозуміти, хто виконує які завдання і як різні частини процесу пов'язані між собою.
- **IDEF (Integration Definition)** – це методика моделювання, яка описує бізнес-процеси на різних рівнях деталізації, що має на меті об'єднання різних частин процесу в одне ціле, створення єдиної системи або структури.

Моделювання передбачає створення діаграм, які показують послідовність дій, відповідальність та зв’язки між різними елементами процесу. Моделі допомагають виявити вузькі місця, затримки або дублювання дій.

Приклад моделі BPMN**:** уявімо процес обробки замовлення клієнта на отримання кредиту. Модель BPMN (рис. 3.7) покаже, як клієнт надсилає запит і

як його обробляє асистент в магазині, і далі передається в кредитний департамент, а потім в фінансовий департамент, і як кінцевий продукт доставляється клієнту.

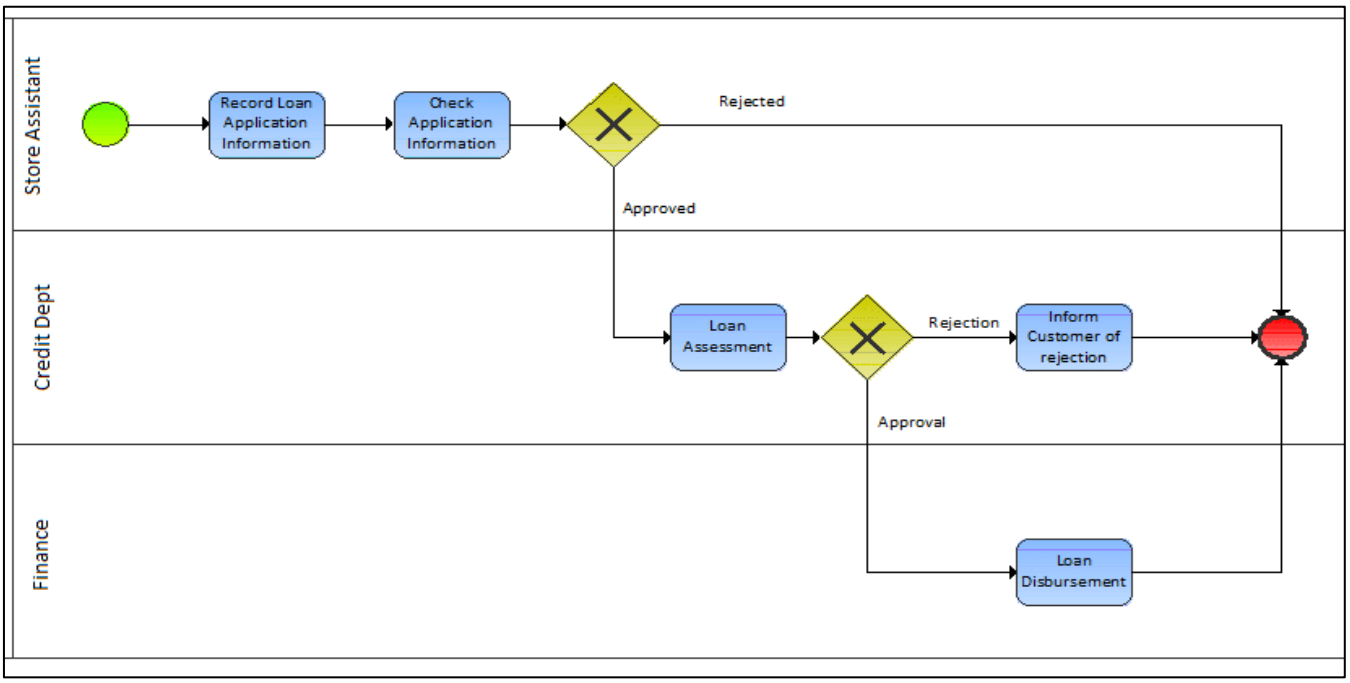

Рисунок 3.6 – Приклад моделі BPMN.

Моделювання бізнес-процесів допомагає оптимізувати роботу організації, *знижуючи витрати та покращуючи ефективність процесів.*

TOGAF-моделі (The Open Group Architecture Framework) – це структура для розробки та управління архітектурою підприємства. Вона використовується для створення стратегії розвитку компанії і допомагає структурувати бізнес-процеси, IT-інфраструктуру та ресурси.

TOGAF допомагає організаціям структурувати свої ресурси та процеси так, щоб досягти стратегічних цілей. Він складається з методів і засобів для розробки архітектури підприємства. Поділяється на чотири домени:

1. *Архітектура бізнесу* – процеси і функції, які необхідні для досягнення цілей організації.
2. *Архітектура даних* – дані, необхідні для підтримки бізнес-процесів.
3. *Архітектура додатків* – IT-системи та програми, що підтримують бізнес.
4. *Технічна архітектура* – інфраструктура та технології, що забезпечують роботу систем.

Етапи TOGAF**:**

1. *Підготовка* – визначення основних вимог та цілей.
2. *Візія* – розробка високорівневої стратегії.
3. *Розробка архітектури* – детальне планування кожного з чотирьох доменів.

4. *Міграція та імплементація* – впровадження змін.

TOGAF є важливим інструментом для управління складними організаціями та їхньою IT-інфраструктурою, забезпечуючи скоординовану та ефективну роботу.

VRIO-аналіз – це стратегічний інструмент для оцінки ресурсів і можливостей організації (рис. 3.7). VRIO означає **V**alue (цінність), **R**arity (рідкість), **I**nimitability (незамінність) та **O**rganization (організація). Аналіз VRIO дозволяє зрозуміти, чи може ресурс або можливість забезпечити довгострокову конкурентну перевагу.

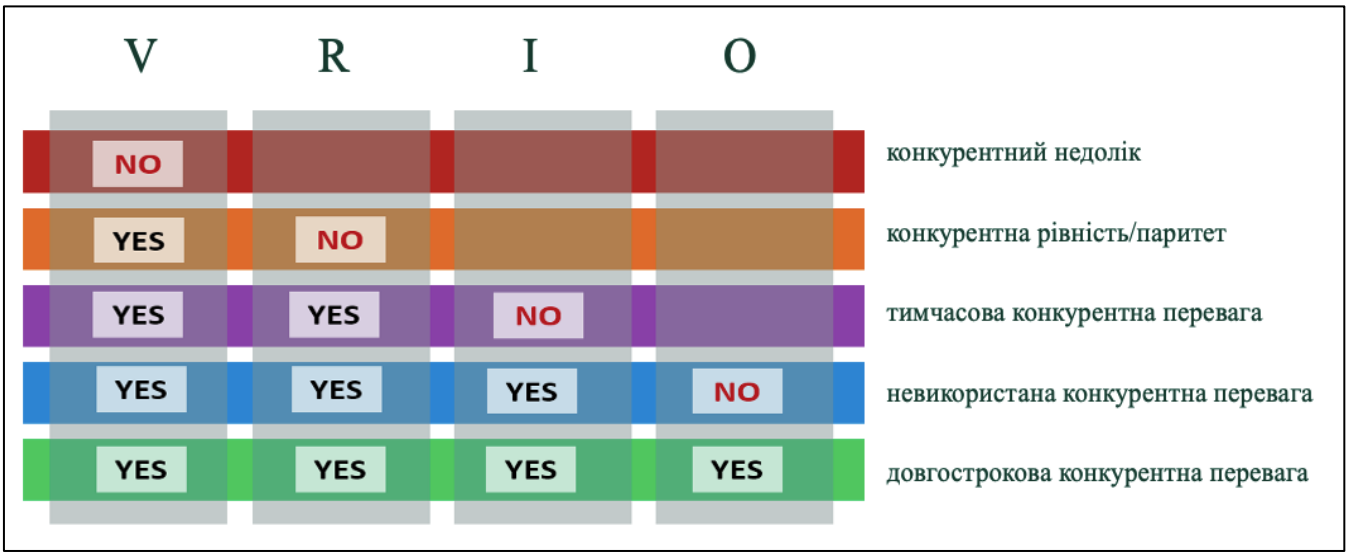

Рисунок 3.7 – Схема VRIO-аналізу.

Елементи VRIO**:**

1. *Цінність (Value):* Чи додає ресурс або можливість цінність для організації? Чи дозволяє він підвищити ефективність, знизити витрати або задовольнити потреби клієнтів?
2. *Рідкість (Rarity):* Чи є цей ресурс або можливість рідкісними? Чим рідкісніший ресурс, тим важче для конкурентів його копіювати.
3. *Незамінність (Inimitability).* Чи можуть конкуренти легко скопіювати або замінити цей ресурс? Якщо ресурс важко імітувати через складність, історичні умови або інновації, він має більшу цінність.
4. *Організація (Organization):* Чи здатна організація ефективно використовувати цей ресурс? Навіть найцінніший ресурс не допоможе, якщо компанія не має відповідної структури або процесів для його ефективного використання.

Використовуючи VRIO-аналіз, компанія може визначити, які її ресурси і можливості є джерелом конкурентної переваги. Наприклад, якщо ресурс є цінним, рідкісним, важко копійованим і компанія має можливість його використовувати, то цей ресурс стає основою для стійкої конкурентної переваги.

Приклад: Компанія, яка володіє унікальною технологією виробництва з низькими витратами (цінність), яку неможливо легко відтворити (незамінність), і має ефективні процеси для її впровадження (організація), може домінувати на ринку.

VRIO-аналіз допомагає організаціям ідентифікувати, які ресурси можуть стати основою для стійкої конкурентної переваги і як їх максимально ефективно використовувати.

Модель бізнес-аналізу Canvas (англ. Business Model Canvas) – це інструмент для візуалізації бізнес-моделі організації (рис. 3.8), перекладається як «полотно». Вона дозволяє зрозуміти, як компанія створює, надає і захоплює цінність.

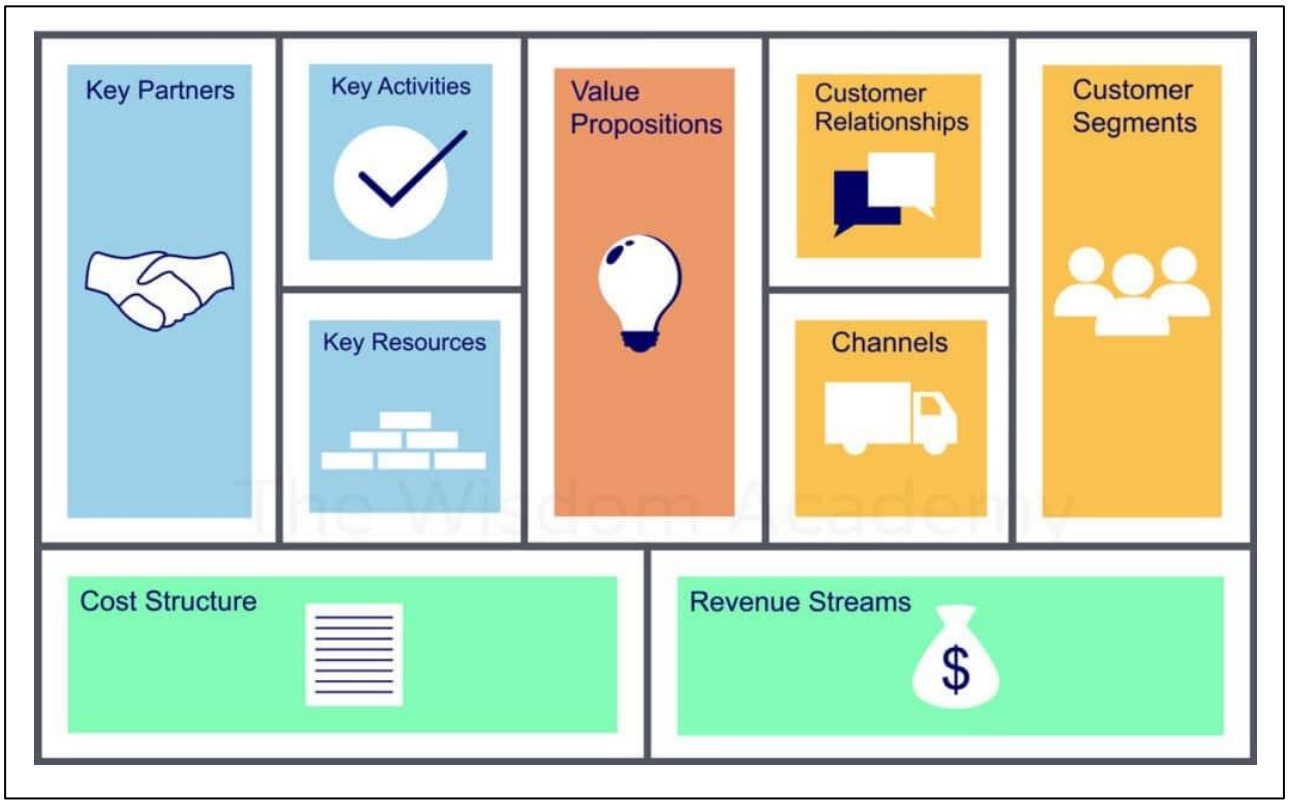

Рисунок 3.8 – Схема бізнес-аналізу Canvas.

Компоненти моделі бізнес-аналізу Canvas**:**

1. **Ключові партнери**. Хто є основними партнерами компанії, які допомагають реалізовувати бізнес-модель?
2. **Ключові види діяльності**. Які основні дії компанія повинна виконувати для досягнення своїх цілей?
3. **Ключові ресурси**. Які ресурси є критичними для функціонування бізнес-моделі?
4. **Ціннісна пропозиція**. Яку цінність компанія пропонує своїм клієнтам? Це може бути продукт або послуга, що задовольняє конкретну потребу.
5. **Взаємовідносини з клієнтами**. Як компанія взаємодіє зі своїми клієнтами на кожному етапі?

6. **Канали збуту**. Як компанія доставляє свою пропозицію клієнтам? Це можуть бути онлайн-канали, фізичні магазини або партнери.
7. **Сегменти клієнтів**. Хто є основними клієнтами компанії? Можливо, це окремі сегменти ринку.
8. **Структура витрат**. Які основні витрати пов'язані з роботою компанії?
9. **Потоки доходів**. Як компанія заробляє гроші? Це можуть бути продажі продуктів, підписки, ліцензування тощо.

Модель бізнесу Canvas дозволяє підприємствам візуалізувати свою бізнес-модель і знайти шляхи для її покращення. Це корисно як для стартапів, які розробляють свої бізнес-моделі з нуля, так і для вже існуючих компаній, що прагнуть оптимізувати свої процеси.

Наприклад: Компанія, що виробляє екологічно чисті продукти, може використовувати модель Canvas, щоб відобразити свої партнерські відносини з постачальниками органічних матеріалів, взаємодію з екологічно свідомими клієнтами та стратегії маркетингу.

Аналіз бізнес-моделі Canvas допомагає структурувати ключові елементи бізнесу та визначити, як організація створює і доставляє цінність своїм клієнтам.

Метод «5 Whys» (*5 Чому*) – це проста (рис. 3.9), але потужна техніка для виявлення кореневої причини проблеми. Вона полягає у тому, щоб п’ять разів запитати «Чому?» щодо проблеми, щоб глибше зрозуміти її першопричину.

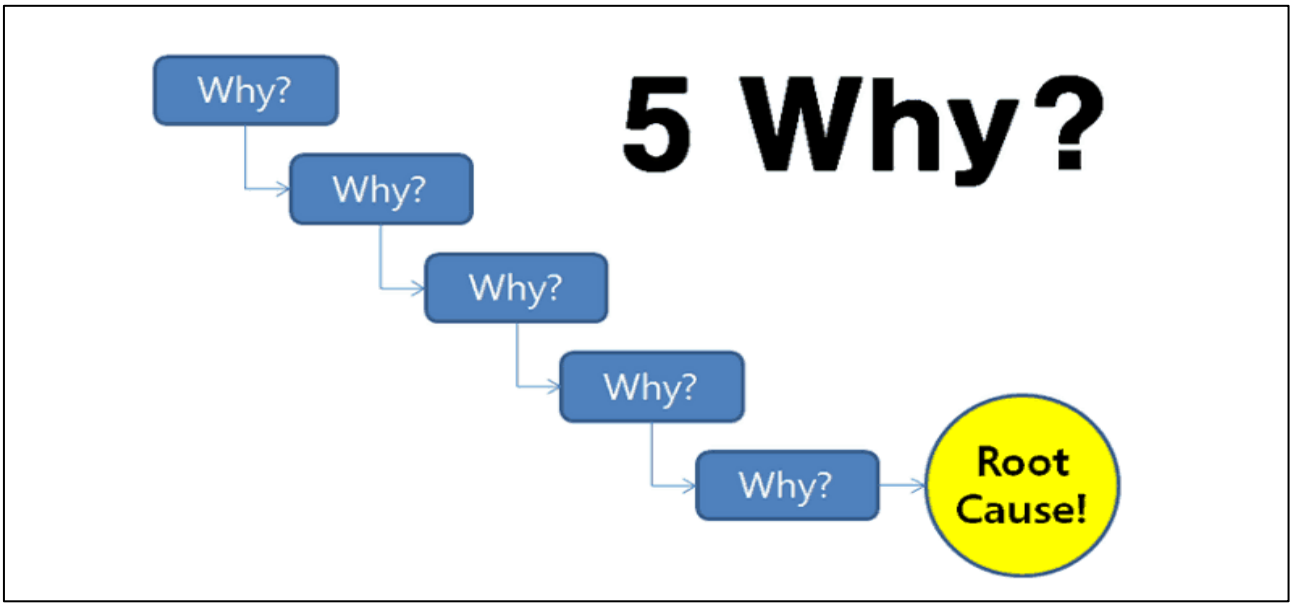

Рисунок 3.9 – Схема методу бізнес-аналізу «5 Whys».

Метод заснований на ітеративному процесі запитання, чому проблема сталася. Кожна відповідь стає відправною точкою для наступного запитання «Чому?», поки не буде виявлена першопричина (англ. Root Cause).

Приклад: Є проблема, яка полягає в тому, що клієнти отримують дефектні продукти:

1. Чому клієнти отримують дефектні продукти? – Через те, що машина виробляє браковані одиниці.
2. Чому машина виробляє браковані одиниці? – Тому що вона налаштована неправильно.
3. Чому вона налаштована неправильно? – Тому що не було проведено належне технічне обслуговування.
4. Чому не було проведено технічне обслуговування? – Тому що у нас немає процесу регулярного обслуговування.
5. Чому немає такого процесу? – Тому що ми не визначили відповідальну особу за це завдання.

Після п’яти «Чому» стає зрозуміло, що корінь проблеми – це відсутність процесу регулярного технічного обслуговування. Виправивши це, компанія зможе уникнути бракованої продукції.

Метод 5 Whys допомагає швидко і ефективно виявляти кореневі причини проблем, що робить його корисним для вирішення як щоденних операційних проблем, так і стратегічних викликів.

Збалансована система показників (Balanced Scorecard, BSC) – це метод управління, що дозволяє організаціям оцінювати свої результати не лише через фінансові показники, але й за іншими ключовими критеріями (рис. 3.10).

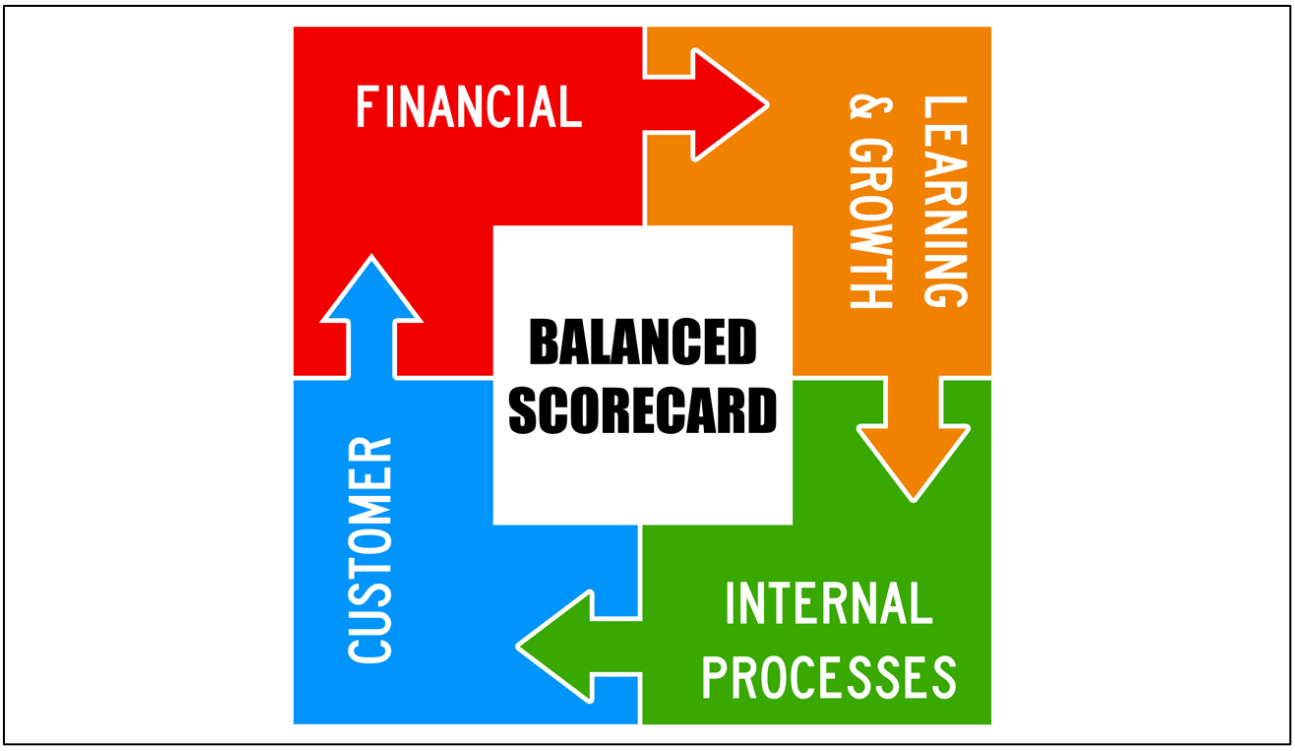

Рисунок 3.10 – Схема методу Збалансованих показників.

Модель була розроблена Робертом Капланом і Девідом Нортоном і включає чотири основні перспективи: фінансову, клієнтську, внутрішні процеси та навчання і розвиток. Розглянемо детально перспективи:

1. *Фінансова перспектива.* Відображає, як компанія виглядає з точки зору фінансових результатів. Включає показники доходу, прибутку, рентабельності інвестицій тощо.
2. *Клієнтська перспектива.* Відображає задоволення потреб клієнтів, ринкову частку, утримання клієнтів та рівень задоволення ними продуктами чи послугами компанії.
3. *Перспектива внутрішніх бізнес-процесів*. Відображає ефективність і якість основних бізнес-процесів, що впливають на створення продукту або надання послуг.
4. *Перспектива навчання і розвитку*. Включає показники розвитку персоналу, інновацій, а також здатність компанії впроваджувати нові технології та підтримувати постійне вдосконалення.

Ця система дозволяє компаніям збалансовано оцінювати свій прогрес за декількома напрямами, замість того, щоб зосереджуватися лише на фінансових результатах. Це сприяє стратегічному управлінню і дозволяє керівництву оцінювати, як різні аспекти бізнесу впливають на досягнення довгострокових цілей.

Наприклад: Компанія може використовувати Balanced Scorecard для вимірювання не лише прибутку, але й рівня задоволення клієнтів, ефективності внутрішніх процесів та рівня навчання персоналу, щоб переконатися, що всі аспекти бізнесу працюють гармонійно.

Balanced Scorecard забезпечує всебічний погляд на діяльність компанії, допомагаючи ефективно керувати бізнесом і досягати стратегічних цілей.

Карта процесів (англ. Process Mapping) – це візуальне представлення бізнес-процесів компанії, яке допомагає зрозуміти, як виконуються різні операції, від початку до кінця. Карта процесів часто використовується для аналізу ефективності, виявлення «вузьких місць» та оптимізації процесів. Основні компоненти карти процесів**:**

1. *Вхідні дані (Inputs)*. Що входить у процес? Це можуть бути ресурси, інформація, матеріали або люди.
2. *Дії та етапи (Activities)*. Які дії виконуються в межах процесу? Це можуть бути завдання, рішення або операції, які ведуть до результату.
3. *Вихідні дані (Outputs)*. Який результат процесу? Це може бути готовий продукт, послуга або інформація.
4. *Зацікавлені сторони*. Хто бере участь у процесі? Це можуть бути працівники, відділи або зовнішні постачальники.

Карта процесів дозволяє візуально відобразити весь цикл діяльності компанії або окремого підрозділу. Це допомагає краще зрозуміти взаємозв’язки між різними етапами і підвищити ефективність за рахунок оптимізації або автоматизації окремих етапів.

Приклад: Карта процесів може показати, як новий клієнт реєструється в системі, як опрацьовується його замовлення і як доставляється продукт. Виявивши, що один з етапів займає занадто багато часу, компанія може вдосконалити процес.

Карта процесів допомагає організаціям чітко зрозуміти свої операції, знайти шляхи для підвищення ефективності та впроваджувати зміни, які покращать бізнес-процеси.

Метод розриву (англ. Gap Analysis) – це метод, який допомагає організаціям виявити розриви між поточними результатами та бажаними цілями (рис. 3.11). Він дозволяє зрозуміти, які кроки потрібно зробити для досягнення запланованих результатів.

	Current State	Future State	Gap	Actions
What	What Happen?	What Happen?	What Happen?	What Happen?
When	When Is It Done?	When Is It Done?	When Is It Done?	When Is It Done?
Where	Where Is Confusion?	Where Is Confusion?	Where Is Confusion?	Where Is Confusion?
Who	Who Will Do It?	Who Will Do It?	Who Will Do It?	Who Will Do It?
How	How It Will Be Solved?	How It Will Be Solved?	How It Will Be Solved?	How It Will Be Solved?

Рисунок 3.11 – Схема методу GAP-аналізу.

Метод складається з чотирьох стадій (етапів), на кожному треба дати відповідь на базові запитання (Що? Коли? Де? Хто? і Як?):

1. *Аналіз поточного стану (англ. Current State)*. Оцінка поточного стану бізнесу або процесу. Які показники досягнуті на сьогоднішній день?
2. *Визначення бажаного стану (англ. Future State)*. Визначення того, які результати компанія хоче досягти в майбутньому.
3. *Виявлення розривів (англ. Gap)*. Порівняння поточного стану з бажаним і визначення розривів між ними.
4. *Розробка плану дій (англ. Actions)*. Визначення, які кроки необхідні для подолання розривів і досягнення бажаних результатів.

Приклад: Компанія може виявити, що її рівень обслуговування клієнтів не відповідає бажаним стандартам. Після проведення GAP-аналізу вона може визначити, що основний розрив виникає через недостатню підготовку персоналу, і розробити план для навчання працівників.

GAP-аналіз дозволяє організаціям чітко визначити, де вони знаходяться, де хочуть бути, і що потрібно зробити для досягнення своїх цілей.

Аналіз рентабельності інвестицій (Return on Investment, ROI) – це метод, який допомагає оцінити ефективність інвестицій, порівнюючи витрати з отриманими вигодами. Цей показник часто використовується для оцінки проектів, маркетингових кампаній та інших ініціатив. Формула для розрахунку ROI**:**

$$\text{ROI} = \frac{\text{Чистий прибуток}}{\text{Вартість інвестицій}} \times 100$$

де, **Чистий прибуток** – це різниця між доходом, отриманим від інвестиції, та її вартістю; **Вартість інвестицій** – це сума, витрачена на реалізацію проекту або інвестиції.

Аналіз ROI дозволяє оцінити, чи була певна інвестиція вигідною, чи досягла очікуваних результатів. Це може бути використано для порівняння різних проектів або ініціатив і вибору найефективніших.

Приклад: Компанія інвестувала $50,000 у нову маркетингову кампанію і отримала додатковий прибуток у розмірі $70,000. За допомогою формули ROI можна розрахувати, що рентабельність інвестицій становить 40%.

Аналіз ROI дозволяє компаніям приймати обґрунтовані рішення щодо інвестицій, оцінюючи їхню ефективність і прибутковість.

Аналіз клієнтських сегментів (англ. Customer Segmentation Analysis) – це процес поділу ринку на окремі групи клієнтів (рис. 3.12), які мають спільні характеристики або потреби. Це дозволяє компаніям краще орієнтуватися на своїх клієнтів і розробляти більш ефективні стратегії маркетингу.

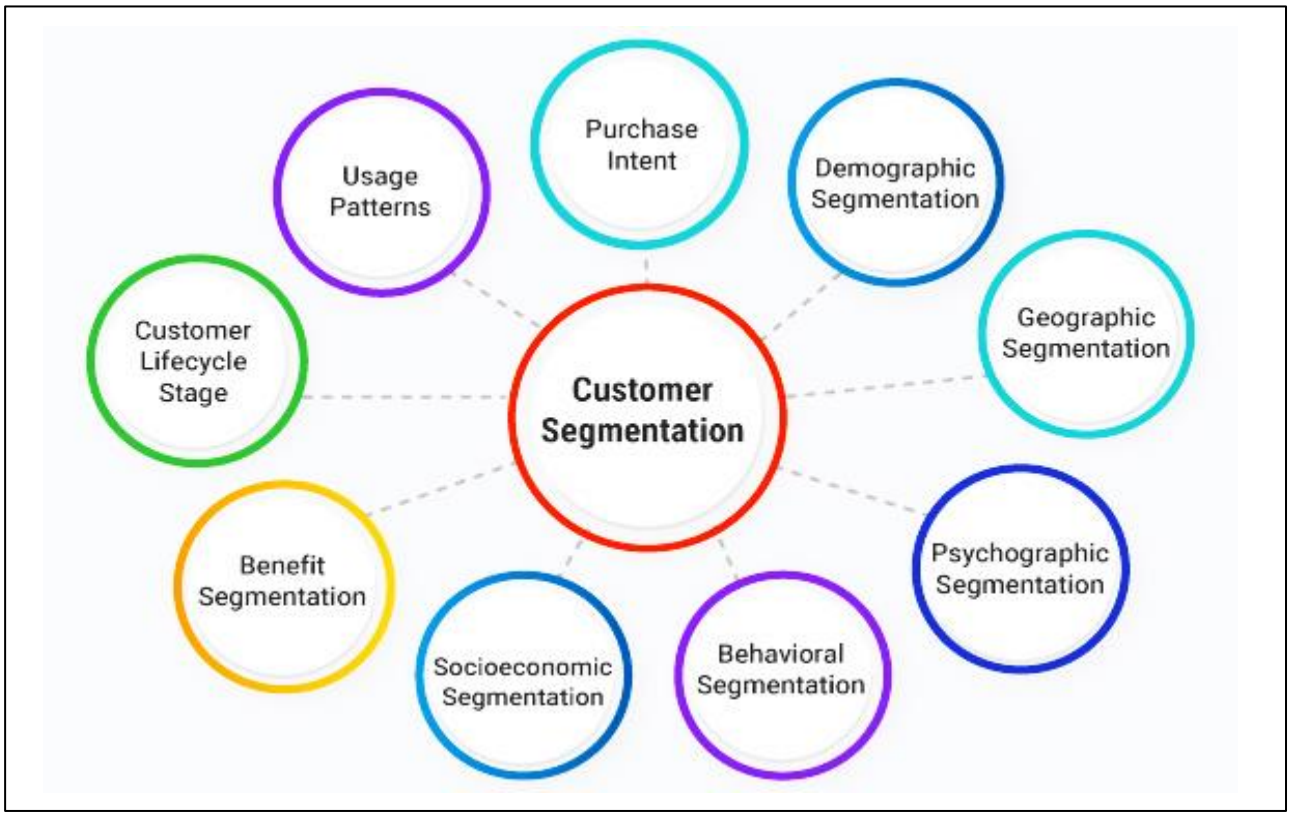

Рисунок 3.12 – Схема методу сегментації ринку.

Основні етапи аналізу сегментації**:**

1. **Визначення критеріїв сегментації**. Це можуть бути демографічні, географічні, психографічні або поведінкові критерії.
2. **Збір даних**. Збір інформації про клієнтів, їхні уподобання, потреби і звички.
3. **Аналіз і групування**. Розподіл клієнтів на сегменти відповідно до визначених критеріїв.
4. **Оцінка сегментів**. Оцінка потенціалу кожного сегмента, його прибутковості і стратегічної важливості.
5. **Розробка стратегій**. Розробка цільових стратегій для кожного сегмента, щоб задовольнити їхні потреби і збільшити продажі.

Приклад: Крім традиційної сегментації на основі віку і доходу, можна використовувати психографічні дані для розподілу клієнтів за їхніми цінностями і стилем життя. Наприклад, компанія може орієнтуватися на екологічно свідомих споживачів для продажу еко-продуктів.

Аналіз клієнтських сегментів дозволяє компаніям краще розуміти своїх клієнтів, адаптувати свою продукцію та послуги до конкретних потреб і максимізувати ефективність маркетингових кампаній.

Ключові показники ефективності (англ. Key Performance Indicators, KPIs) – це вимірювальні показники, які дозволяють оцінювати досягнення стратегічних і операційних цілей організації. Типи KPIs**:**

1. *Фінансові KPIs*. Включають показники, такі як прибуток, рентабельність, оборот капіталу, ROI.
2. *Операційні KPIs*. Оцінюють ефективність внутрішніх процесів, такі як швидкість обробки замовлень, рівень дефектів продукції.
3. *Клієнтські KPIs*. Вимірюють задоволення клієнтів, утримання клієнтів, ринкову частку.
4. *HR KPIs*. Оцінюють ефективність управління людськими ресурсами, такі як рівень утримання співробітників, продуктивність працівників.

Першим кроком є визначення стратегічних цілей компанії. Далі, для кожної цілі потрібно встановити відповідні KPIs, які будуть використовуватися для вимірювання успіху. Регулярний моніторинг і аналіз KPIs дозволяє відстежувати прогрес і вчасно вживати коригувальних заходів.

Якщо стратегічною метою компанії є підвищення задоволення клієнтів, то KPI може включати показник рівня задоволеності клієнтів через опитування або спосіб вимірювання.

KPIs допомагають організаціям визначити, наскільки успішно вони досягають своїх цілей, і сприяють прийняттю обґрунтованих рішень для покращення результатів.

Аналіз впливу (англ. Impact Analysis) – це процес оцінки наслідків змін або нововведень на різні аспекти бізнесу чи проекту. Цей метод дозволяє зрозуміти, як зміни вплинуть на існуючі процеси, ресурси, стейкхолдерів та результати. Основні етапи аналізу впливу**:**

1. *Визначення змін*. Чітке формулювання змін або нововведень, які потрібно оцінити.
2. *Оцінка можливих наслідків*. Аналіз того, як ці зміни можуть вплинути на різні аспекти бізнесу, такі як процеси, фінанси, людські ресурси.
3. *Ідентифікація зацікавлених сторін*. Визначення, які стейкхолдери будуть найбільше впливати або будуть під впливом змін.
4. *Розробка плану управління впливом*. Розробка стратегії для управління негативними наслідками та використання позитивних впливів.

Приклад: Якщо компанія вирішує впровадити нову систему управління, аналіз впливу може включати оцінку, як це вплине на існуючі IT-процеси, навчання персоналу і загальний бюджет проекту. Аналіз впливу допомагає організаціям передбачити наслідки.

Аналіз витрат і вигод (англ. Cost-Benefit Analysis) – це метод оцінки економічної вигоди від певного проекту або рішення шляхом порівняння його витрат з отриманими вигодами. Це допомагає визначити, чи є проект чи рішення фінансово обґрунтованим. Етапи аналізу витрат і вигод**:**

1. *Визначення витрат*. Оцінка всіх витрат, пов'язаних із проектом або рішенням, включаючи прямі і непрямі витрати.
2. *Оцінка вигод*. Визначення всіх вигод, які будуть отримані в результаті реалізації проекту або рішення.
3. *Порівняння витрат і вигод*. Аналіз співвідношення витрат до вигод, щоб оцінити, чи виправдовують вигоди витрати.
4. *Прийняття рішення*. Прийняття рішення на основі результатів аналізу витрат і вигод.

Приклад: Перед впровадженням нової системи автоматизації, компанія може провести аналіз витрат і вигод, щоб зрозуміти, чи зменшить система витрати на робочу силу і збільшить продуктивність більше, ніж буде витрачено на впровадження системи.

Аналіз витрат і вигод допомагає оцінити економічну доцільність проектів і рішень, забезпечуючи обґрунтованість інвестицій і ресурсів.

Юзерсторі (англ. User Stories) – це простий і ефективний метод опису вимог до функціональності системи або продукту з точки зору кінцевого користувача. Вони використовуються для визначення того, яку цінність користувач отримає від певної функції. Юзерсторі допомагають чітко сформулювати потреби користувачів і забезпечують спільну мову для команди розробників, бізнес-аналітиків і стейкхолдерів.

Структура юзерсторі дуже проста і складається із трьох компонентів:

1. *Роль користувача*: Хто є користувачем? Наприклад, «клієнт» або «менеджер».
2. *Бажання чи потреба*. Що саме користувач хоче зробити? Наприклад, «я хочу зробити покупку онлайн» або «відправити запит».
3. *Цінність*. Яка користь від цього? Наприклад, «щоб зекономити час і уникнути черг» або «зменшити витрати».

Типова форма юзерсторі виглядає так:

«Я, як **[користувач]**, хочу **[дія]**, щоб **[цінність]**»

Деталі юзерсторі**:**

- *Критерії прийняття (англ. Acceptance Criteria)*. Це чіткі умови, які повинні бути виконані, щоб історія вважалася завершеною. Вони описують очікувану поведінку системи.
- *Обговорення і завдання*. Юзерсторі слугує точкою початку для обговорення функціональності. Команда може ділити історію на завдання для різних членів команди (наприклад, дизайн, розробка, тестування).

Приклад юзерсторі: «Я, як покупець, хочу мати можливість додати товар до кошика, щоб я міг продовжити вибір товарів і зберегти свій вибір для покупки пізніше.»

Критерії прийняття для цієї юзерсторі:

1. Товар додається до кошика одним кліком.
2. Кошик зберігає товар, навіть якщо користувач виходить з системи.
3. Користувач може переглядати кошик і видаляти з нього товари.

Юзерсторі допомагають командам чітко розуміти, що хочуть користувачі, і забезпечують фокус на кінцевій цінності для користувача. Це дозволяє створювати продукти, які відповідають реальним потребам і покращують взаємодію з системою.

4. ДОКУМЕНТИ ПРОЄКТУ

4.1 Збір вимог

Документування проектів це ключовий процес у життєвому циклі розробки будь-якого продукту, від невеликого стартапу до масштабних корпоративних рішень. Проектна документація допомагає структуровано зафіксувати всі аспекти роботи над проектом, від планування до реалізації, і є важливим інструментом для управління командою, ресурсами та термінами.

Уявімо, що ви вирушаєте у складну подорож без карти, де кожен крок потребує глибокого планування та чітких вказівок. Точно так само будь-який проект без належної документації може швидко перетворитися на хаотичний процес, сповнений непорозумінь, невідповідностей та помилок. Документи проекту, по суті, є тією «картою», яка допомагає команді рухатися у правильному напрямку, встановлює чіткі правила, визначає завдання та відповідає за ефективну комунікацію між усіма учасниками.

Цей розділ присвячений розгляду основних видів проектної документації, її значенню та впливу на успішність проекту. Ми детально розберемося з тим, які саме документи створюються на кожному етапі життєвого циклу проекту, хто відповідає за їх підготовку, які стандарти та вимоги слід враховувати під час їх розробки, а також як правильно організувати документообіг для забезпечення ефективної взаємодії між членами команди та іншими зацікавленими сторонами.

У кожному проекті, незалежно від його масштабу чи галузі, є багато аспектів, які потрібно враховувати: цілі проекту, вимоги, технічні специфікації, плани, ризики, оцінка ресурсів і бюджету. Без правильної документації ці аспекти можуть бути незрозумілими або залишитися непоміченими, що може призвести до помилок, затримок або навіть провалу проекту. Документи проекту виконують низку важливих функцій:

Засіб комунікації: Документація проекту забезпечує ефективний обмін інформацією між різними учасниками проекту, включаючи розробників, дизайнерів, аналітиків, менеджерів та замовників. Вона дозволяє всім членам команди мати спільне розуміння цілей, завдань та вимог проекту, що особливо важливо у великих командах або при співпраці з зовнішніми підрядниками.

Основа для прийняття рішень: Завдяки правильно складеним документам, менеджери проектів можуть приймати обґрунтовані рішення щодо пріоритетів, розподілу ресурсів, визначення ризиків та вирішення проблем. Документація дає змогу оцінювати прогрес проекту, відстежувати виконання завдань та коригувати стратегії у разі необхідності.

Юридична захищеність: У багатьох випадках проектна документація має юридичне значення. Контракти, технічні завдання, специфікації та акти приймання-передачі – це ті документи, які юридично зобов’язують сторони виконувати певні зобов’язання. Наявність таких документів допомагає уникнути суперечок і вирішувати конфліктні ситуації у правовому полі.

Збереження знань: Документи проекту часто служать базою знань для майбутніх проектів. Вони допомагають новим членам команди швидко вникнути в проект, зрозуміти його особливості, вимоги та досягнуті результати. Крім того, документація дозволяє уникати повторення помилок та використовувати кращі практики для майбутніх проектів.

Контроль та аудит: Проектна документація відіграє важливу роль у контролі за виконанням завдань та оцінці результатів проекту. Вона дозволяє стежити за виконанням плану, бюджетом та термінами. У разі аудиту проекту, документи можуть слугувати доказом виконаної роботи та дотримання встановлених вимог.

У рамках проекту створюються різні типи документів, кожен з яких має своє призначення та зміст. Документи можна розділити на кілька категорій, відповідно до етапів життєвого циклу проекту:

Ініціаційна документація: Ці документи створюються на початковій стадії проекту і містять ключову інформацію про його цілі, масштаби та загальні вимоги. До таких документів належать бізнес-кейс, статут проекту, попередні оцінки бюджету та графік виконання. Основне завдання цього типу документації – отримати згоду всіх зацікавлених сторін щодо того, чого має досягти проект.

Планувальна документація: На цьому етапі створюються більш детальні документи, що визначають, як буде реалізований проект. Це плани управління проектом, ризиками, якістю, ресурсами, а також специфікації, що описують технічні вимоги до продукту чи послуги. Планувальна документація дозволяє команді чітко розуміти, як саме буде виконуватися робота, і що потрібно для досягнення успіху.

Операційна документація: Це документи, що описують робочі процеси та виконані завдання під час реалізації проекту. Вони можуть включати звіти про виконання роботи, оновлені плани та журнали ризиків, що виникають протягом проекту. Ця документація дозволяє відстежувати поточний статус проекту та коригувати плани у разі змін.

Заключна документація: Після завершення проекту створюється заключна документація, яка підсумовує результати роботи та оцінює досягнення цілей. Вона включає акти приймання-передачі, фінальні звіти, аналітичні огляди та уроки, які були вивчені під час реалізації проекту. Заключна документація є важливим джерелом знань для майбутніх проектів та доказом виконаної роботи для замовників або керівництва.

Однією з ключових особливостей розробки проектної документації є дотримання стандартів, що регламентують процеси управління проєктами та документообіг. Ці стандарти встановлюють вимоги до структури документів, їх оформлення та змісту. Дотримання стандартів дозволяє забезпечити уніфікований підхід до документації, підвищити її якість та полегшити взаємодію між різними командами та зовнішніми підрядниками.

Після того як ініціатор подав проектну пропозицію (Business Case) та сформував робочу групу (Команду проекту) слід зібрати вимоги до проекту від усіх зацікавлених сторін (Stakeholders). На цьому етапі починає активно працювати бізнес-аналітик.

Згадаємо, що *бізнес-аналіз* – це процес дослідження потреб організації або проекту з метою розробки ефективних рішень. Він фокусується на виявленні проблем і можливостей, формуванні вимог, оптимізації бізнес-процесів та забезпеченні інтеграції нових рішень у бізнес-структуру.

Бізнес-аналіз передбачає збирання, дослідження та обробку даних про організацію, її ринок, конкурентів, процеси і потреби. Основною метою є пошук оптимальних шляхів для досягнення поставлених цілей, покращення продуктивності або адаптації до змін.

Бізнес-аналітик – посередником між бізнесом і технічними командами допомагає зрозуміти потреби бізнесу та перетворити їх на конкретні вимоги для розробників проводячи процес бізнес-аналізу.

На практиці, бізнес-аналітик розробляє документ, який надсилає (презентує) всім зацікавленим сторонам проекту із запитом заповнити його (рис. 4.1). Він розробляє його спираючись на основні потреби проекту, цілі та задачі, а також на можливі вимоги ринку. Зазвичай цей документ має формат брифу (англ. Brief) і носить таку саму назву як і сам процес – Requirements Gathering Brief (Бриф для збору вимог).

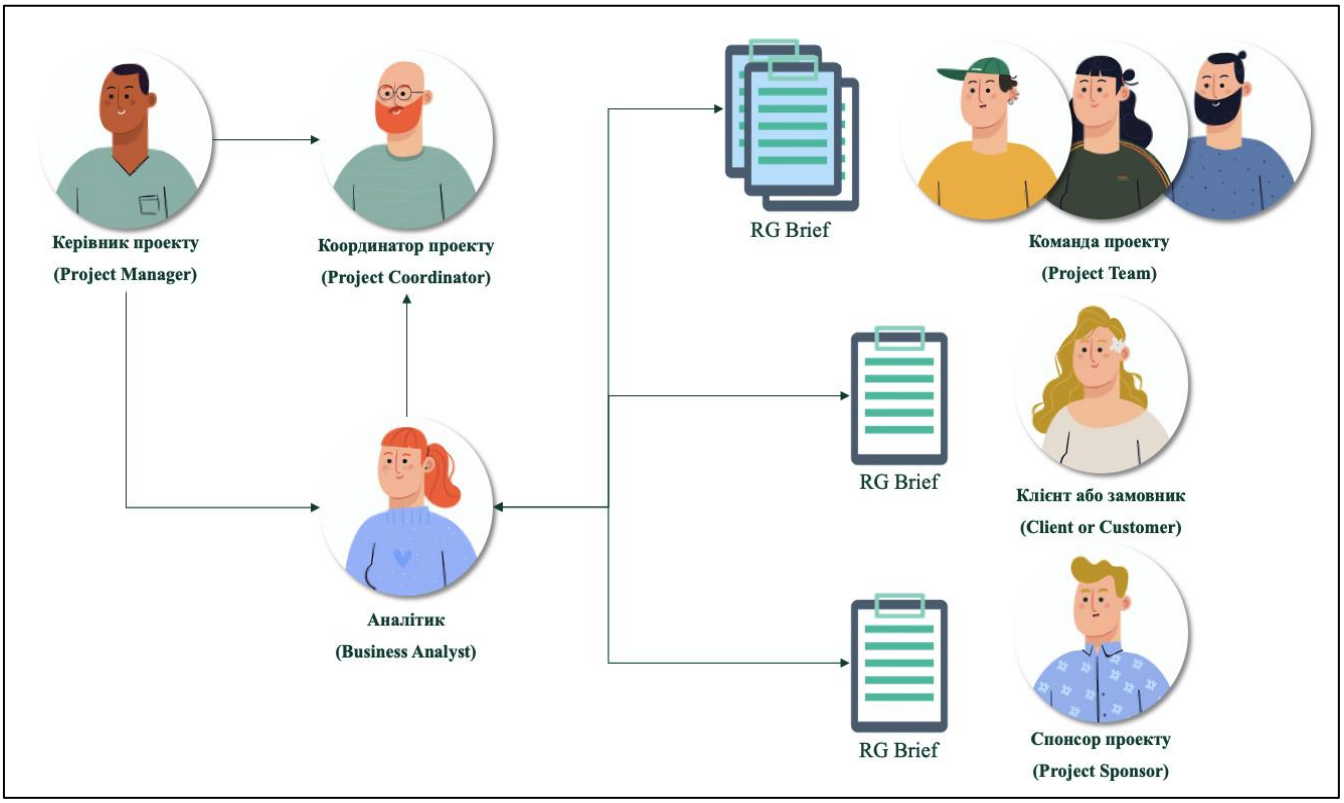

Рисунок 4.1 – Процес збору вимог у зацікавлених сторін.

Збір вимог (англ. Requirements Gathering) – це процес виявлення, визначення, документування та аналізу вимог, необхідних для створення продукту чи системи. Метою цього процесу є отримання чіткого розуміння того, що має бути реалізовано, щоб задовольнити потреби користувачів та бізнесу. Структура документа збору вимог:

1. **Вступ**: Опис проекту, Основна мета системи або продукту, Контекст та ключові терміни.
2. **Зовнішні вимоги**: Очікування від зацікавлених сторін, Інтеграція з іншими системами або зовнішніми компонентами.
3. **Функціональні вимоги**: Опис функцій, які система має виконувати, (Приклади: обробка замовлень, управління користувачами тощо).
4. **Нефункціональні вимоги**: Опис характеристик, які стосуються якості системи. (Приклади: продуктивність, безпека, масштабованість, доступність).
5. **Обмеження та припущення**: Будь-які обмеження проекту, наприклад, технологічні, бюджетні, часові, Припущення, зроблені під час визначення вимог.
6. **Сценарії використання (Use Cases)**: Опис конкретних сценаріїв взаємодії користувачів із системою.
7. **Пріоритизація вимог**: Розподіл вимог за пріоритетністю (обов'язкові, бажані, опціональні).
8. **Критерії прийняття**: Умови, за яких система або продукт вважаються завершеними та прийнятими.

Цей документ забезпечує вирішення ключових потреб на цьому етапі:

Комунікація: Він забезпечує чітке розуміння між замовником, розробниками та іншими зацікавленими сторонами щодо того, що потрібно реалізувати.

Основа для планування: Вимоги є основою для складання технічних завдань, оцінки часу та ресурсів.

Запобігання помилкам: Документ допомагає уникнути непорозумінь і неузгодженостей між учасниками проекту.

Контроль: У процесі реалізації проекту документ може використовуватися для перевірки відповідності продукту початковим вимогам.

Таким чином, збір вимог є ключовим етапом, який значно впливає на успіх проекту, оскільки забезпечує чітке розуміння того, що необхідно реалізувати, за допомогою технік, методів та інструментів бізнес-аналізу.

4.2 Запит пропозиції

Запит на пропозицію (англ. Request for Proposal, RFP) – це документ, який організації використовують для залучення постачальників або підрядників до

подання комерційних пропозицій на виконання конкретного проекту або надання послуг. RFP дозволяє організації отримати детальну інформацію про можливості постачальників, їхні пропозиції, ціни та інші важливі аспекти. Розглянемо рекомендовану структуру документу RFP**:**

1. **Вступ**:
   - Опис організації.
   - Мета запиту (чому потрібен RFP).
2. **Опис проекту**:
   - Деталі про проект, включаючи його мету, обсяги та цілі.
3. **Вимоги до пропозиції**:
   - Функціональні та нефункціональні вимоги.
   - Очікування від постачальників.
4. **Критерії відбору**:
   - Критерії, за якими буде оцінюватися пропозиція (ціна, досвід, технології тощо).
5. **Терміни**:
   - Дати, коли пропозиції мають бути подані, та інші важливі терміни.
6. **Фінансові умови**:
   - Інформація про бюджет, спосіб оплати та умови фінансування.
7. **Процес подання пропозицій**:
   - Інструкції для постачальників про те, як подати свої пропозиції.
   - Формат, в якому повинні бути подані документи.
8. **Запитання та відповіді**:
   - Процедура, як постачальники можуть ставити запитання щодо RFP та отримувати відповіді.
9. **Додатки**:
   - Додаткова інформація, яка може бути корисною для постачальників (наприклад, технічні специфікації, плани проекту).

Для чого потрібен RFP**:**

Стандартизація процесу: RFP допомагає структурувати процес залучення постачальників, що дозволяє спростити порівняння пропозицій.

Залучення різноманітних пропозицій: Через RFP організація може отримати різні варіанти рішень від кількох постачальників, що дозволяє вибрати найкращий варіант.

Управління ризиками: Підготовка RFP змушує організацію детально подумати про свої вимоги та цілі, що може зменшити ризики у проекті.

Покращення комунікації: Документ служить основою для спілкування між організацією та постачальниками, що допомагає уникнути непорозумінь.

Прозорість і конкуренція: Використання RFP підвищує прозорість процесу вибору постачальника і стимулює конкуренцію, що може призвести до кращих умов та цін.

Таким чином, RFP є важливим інструментом для організацій, які шукають ефективні рішення для своїх потреб, забезпечуючи структуру та порядок у процесі вибору постачальників.

4.3 Комерційна пропозиція

Комерційна пропозиція (англ. Commercial Proposal) – це документ, який підрядники або постачальники послуг використовують для представлення своїх продуктів або послуг потенційним клієнтам. Основна мета комерційної пропозиції – переконати клієнта обрати їхнє рішення або продукт, надаючи детальну інформацію про те, що пропонується, і чому це вигідно для замовника. Приклад структури комерційної пропозиції:

1. **Вступ**:
   - Короткий опис компанії.
   - Мета пропозиції та її значення для клієнта.
2. **Опис продукту/послуги**:
   - Детальний опис товарів або послуг, які пропонуються.
   - Включення характеристик, переваг і унікальних особливостей.
3. **Цінова пропозиція**:
   - Чітка інформація про ціни, знижки, акції та умови оплати.
   - Можливі варіанти тарифів або пакетів.
4. **Терміни виконання**:
   - Часові рамки для виконання замовлення або реалізації проекту.
   - Визначення етапів, якщо це необхідно.
5. **Вимоги та умови**:
   - Умови виконання, включаючи відповідальність сторін.
   - Інформація про гарантії та післяпродажне обслуговування.
6. **Досвід і відгуки**:
   - Інформація про попередні проекти, досвід роботи.
   - Відгуки або рекомендації від інших клієнтів.

7. **Заключення**:
   - Запрошення до співпраці.
   - Контактна інформація для подальшого спілкування.

Для чого потрібна комерційна пропозиція:

Презентація рішення: Комерційна пропозиція дозволяє компаніям продемонструвати свої рішення і переконати клієнтів у їхній цінності.

Конкуренція: Вона допомагає компаніям виділитися серед конкурентів, підкреслюючи свої унікальні пропозиції та переваги.

Установлення зв'язку з клієнтом: Пропозиція служить базою для подальшого спілкування між постачальником і потенційним клієнтом, сприяючи встановленню довіри.

Управління очікуваннями: Чітко визначаючи умови та вимоги, комерційна пропозиція допомагає уникнути непорозумінь під час реалізації проекту.

Підтримка ухвалення рішень: Документ надає потенційним клієнтам всю необхідну інформацію для ухвалення обґрунтованого рішення про покупку.

Таким чином, комерційна пропозиція є важливим інструментом для бізнесу, що дозволяє ефективно взаємодіяти з клієнтами та підвищувати ймовірність укладення угод.

Отримавши комерційні пропозиції від потенційних постачальників бізнес-аналітик на наступному етапі створює зведену таблицю, в якій і порівнює ключові відповіді. Конкретний стандартний шаблон документу на цьому етапі відсутній, але зазвичай, це зведена таблиця та звіт (презентація).

Для порівняння пропозицій, частіше за все використовують методи вагових коефіцієнтів та питомих показників.

Порівняння комерційних пропозицій на програмне забезпечення методом вагових коефіцієнтів є ефективним способом систематизувати оцінку різних пропозицій на основі визначених критеріїв. Цей метод дозволяє структурувати процес ухвалення рішень і допомагає вибрати найбільш підходящий варіант. Розглянемо покроковий підхід до цього процесу.

Почнемо з визначення критеріїв оцінки. Якщо ми аналізуємо складну систему вимог, що має кілька вкладених рівнів, то варто пропорційно розподілити їх вплив на всю оцінку.

Спочатку необхідно визначити ключові критерії, за якими будуть оцінюватися комерційні пропозиції. Приклади критеріїв можуть включати:

1. Функціональність
2. Ціна
3. Технічна підтримка
4. Досвід постачальника

Далі, встановлення вагових коефіцієнтів для кожного, який відображає його важливість у загальному оцінюванні. Вагові коефіцієнти можуть бути розраховані у відсотках або в балах, при цьому сума всіх вагових коефіцієнтів повинна дорівнювати 1 або 100%. Наприклад:

1. Функціональність: 40%
2. Ціна: 30%
3. Технічна підтримка: 15%
4. Досвід постачальника: 15%

Наступним кроком ми оцінюємо кожну комерційну пропозицію за кожним критерієм на шкалі, наприклад, від 1 до 5 або від 1 до 10, де 1 – це найгірший результат, а 5 (або 10) – найкращий.

Оцінивши кожну пропозицію слід обчисліть загальну оцінку, помноживш отриману оцінку за кожним критерієм на відповідний ваговий коефіцієнт і потім підсумувавши ці значення:

$$\text{Загальна оцінка} = \sum \text{Оцінка за критерієм} \times \text{Ваговий коефіцієнт}$$

Розглянемо приклад, в якому ми отримали три комерційні пропозиції (A,B та C) від потенційних постачальників програмних продуктів (табл. 4.1).

Таблиця 4.1 – Приклад оцінки комерційних пропозицій

Критерій	**Ваговий коефіцієнт**	**Пропозиція A**	**Пропозиція B**	**Пропозиція C**
Функціональність	0.4	4	5	3
Ціна	0.3	3	4	5
Технічна підтримка	0.15	5	4	2
Досвід постачальника	0.15	3	5	4

Тепер, користуючись формулою, вирахуємо оцінку кожної пропозиції:

Пропозиція A: (4×0.4)+(3×0.3)+(5×0.15)+(3×0.15) = 3.2+0.9+0.75+0.45 = 5.3

Пропозиція B: (5×0.4)+(4×0.3)+(4×0.15)+(5×0.15) = 2.0+1.2+0.6+0.75 = **5.55**

Пропозиція C: (3×0.4)+(5×0.3)+(2×0.15)+(4×0.15) = 1.2+1.5+0.3+0.6 = 3.6

Порівнявши, бачимо, що Пропозиція B найкраща, й найбільше задовольняє наші вимоги та очікування. Звісно, прийняття рішення може базуватись на великій кількості факторів, а таке оцінювання і порівняння відображає

суб’єктивну оцінку, хоч і базується на математичному інструменті. Та такий підхід допомагає в обробці великої кількості пропозицій із багаточисленними факторами.

4.4 Замовлення на покупку

Замовлення на покупку (англ. Purchase Order, PO) – це офіційний документ, який покупець надсилає постачальнику для підтвердження замовлення на товари або послуги. Це важливий елемент процесу закупівель, який забезпечує чіткість і організацію у взаємодії між покупцем і постачальником.

Даний документ є додатком до основного договору між Постачальником і Замовником, зазвичай.

Постачальник і замовник укладають головний (рамковий) договір, в якому домовляються про умови співпраці, свої права та обов’язки. А окремі покупки визначають додатками до цього договору, які є невід’ємною його частиною. Це дуже зручно в контексті розмежування між різними проектами, продуктами та послугами. Оскільки один постачальник може надавити багато послуг різним командами (департаментам) замовника, і такі проекти повинні бути незалежні фінансово та організаційно.

До того ж, саме така організація допомагає реалізувати методологію гнучкості управління проектів. Наприклад, виділяти в окремі замовлення різні релізи продуктів із відповідним набором очікуваних функцій.

Додатками до кожного замовлення є технічне завдання (SoW), специфікації, тощо. На рис 5.1 наведено принципову схему зв’язку документів.

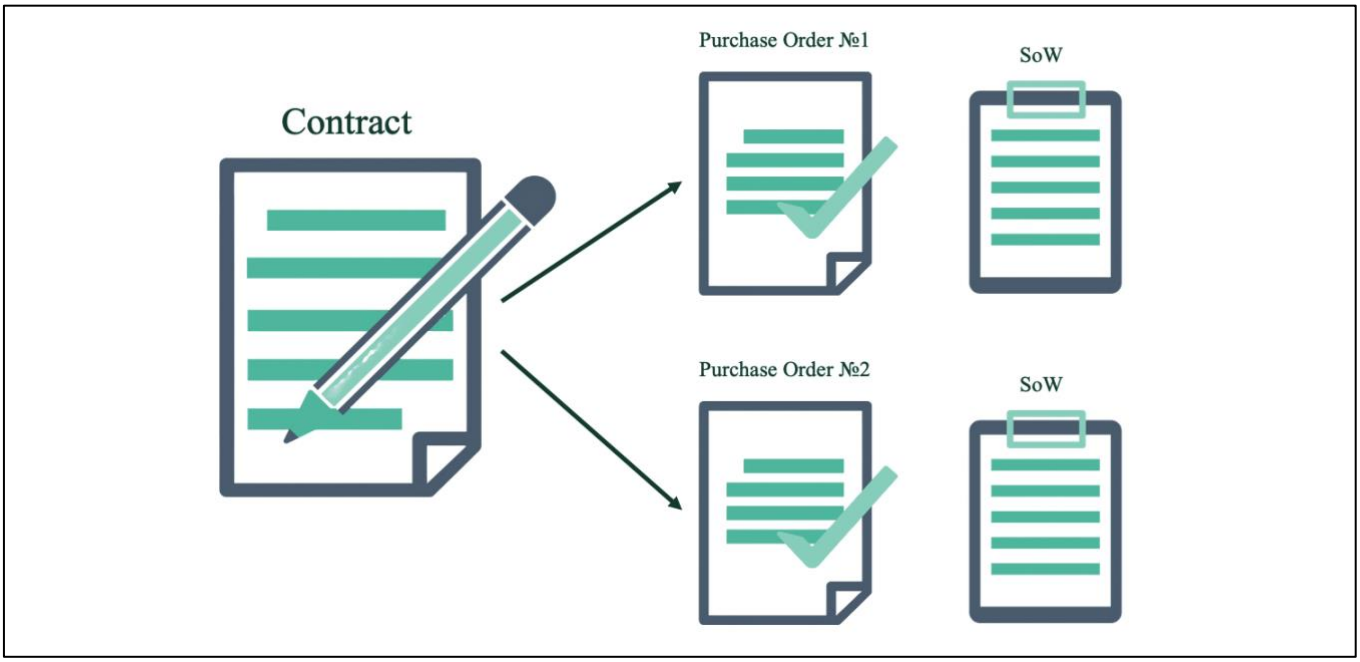

Рисунок 4.2 – Замовлення на покупку в структурі документів.

Замовлення на покупку обов’язково повинно містити наступну інформацію та розділи**:**

1. **Заголовок документа**:
   - Назва документа (Purchase Order).
   - Номер замовлення (унікальний ідентифікатор для відстеження).
2. **Дані про покупця**:
   - Назва компанії покупця.
   - Адреса.
   - Контактна особа (ім'я, телефон, електронна пошта).
3. **Дані про постачальника**:
   - Назва компанії постачальника.
   - Адреса.
   - Контактна особа (ім'я, телефон, електронна пошта).
4. **Дата замовлення**:
   - Дата, коли було створено замовлення.
5. **Опис товарів або послуг**:
   - Перелік товарів або послуг, які замовляються.
   - Кількість.
   - Одиниця виміру (наприклад, штуки, метри).
   - Ціна за одиницю.
6. **Умови доставки**:
   - Місце доставки.
   - Дата або терміни доставки.
   - Умови доставки.
7. **Умови оплати**:
   - Умови та методи оплати (наприклад, банківський переказ, післяплата, передоплата, кілька платежів).
   - Термін оплати (наприклад, 30 днів після отримання).
8. **Додаткова інформація**:
   - Будь-які примітки або спеціальні вимоги.
   - Посилання на попередні угоди або контракти, якщо це необхідно.
9. **Підпис**:
   - Підписи уповноважених осіб.

Для чого потрібне замовлення на покупку:
Офіційний документ: РО служить юридично обґрунтованим документом, який підтверджує угоду між покупцем і постачальником.

Запобігання помилкам: Чітке викладення вимог допомагає уникнути непорозумінь щодо замовлених товарів або послуг.

Контроль витрат: Замовлення на покупку допомагає підприємству контролювати витрати, оскільки всі замовлення документуються.

Відстеження замовлень: Наявність унікального номера замовлення спрощує відстеження та управління замовленнями.

Упорядкування бухгалтерії: Замовлення на покупку полегшує бухгалтерський облік, оскільки зв'язує рахунки-фактури з конкретними замовленнями.

Стандартізація процесів: Використання замовлень на покупку сприяє стандартизації процесів закупівель у компанії, що полегшує їхнє управління.

Отже, замовлення на покупку є важливим інструментом для управління процесом закупівель, забезпечуючи чіткість, контроль та ефективність у взаємодії між покупцями та постачальниками.

4.5 Технічне завдання

Технічне завдання **(***ТЗ або англ. Statement of Work, SoW***)** – це ключовий документ, що визначає рамки та вимоги до проекту, продукту або системи. Його можна вважати своєрідною інструкцією, яка детально описує, що потрібно зробити, які технології використати, яким має бути кінцевий результат, і як цей результат оцінювати. Для замовників та розробників ТЗ є фундаментом, на якому будується проект. Неправильно складене або неповне ТЗ може призвести до нерозуміння між сторонами, затримок у виконанні та навіть провалу проекту.

Технічне завдання є особливо важливим у сфері інформаційних технологій, де вимоги до програмного забезпечення, мобільних додатків або вебсайтів повинні бути чітко визначені та відповідати очікуванням замовника і користувачів. Розглянемо основні елементи технічного завдання:

Мета проекту – це чітке формулювання завдань, які має вирішити проект, і того, що замовник очікує від результату. Важливо, щоб ця частина ТЗ була максимально конкретною та зрозумілою для всіх учасників проекту. Мета проекту може включати створення нового продукту, модернізацію існуючого або вирішення певної бізнес-проблеми.

Опис проекту – це загальний огляд проекту, який включає основні функції, завдання та очікувані результати. Наприклад, якщо це ТЗ на розробку вебсайту, опис проекту включатиме інформацію про функціональність сайту, цільову аудиторію, необхідні інтеграції та технології.

Функціональні вимоги – це перелік конкретних функцій, які повинні бути реалізовані в рамках проекту. Функціональні вимоги визначають, як система має працювати та яку поведінку очікують від неї користувачі. Для програмного забезпечення це може включати такі функції, як авторизація користувачів, управління контентом, робота з базами даних тощо.

Технічні вимоги – це специфікації до інфраструктури, мов програмування, платформ, баз даних та інших технологій, які використовуватимуться під час реалізації проекту. У технічних вимогах може бути зазначено, на якій операційній системі буде працювати програма (наприклад, Windows, macOS або Linux), який стек технологій буде використано (наприклад, Python, Java, JavaScript), а також вимоги до продуктивності системи, надійності та безпеки.

Нефункціональні вимоги – це вимоги, які стосуються продуктивності системи, її надійності, масштабованості та зручності користування. Нефункціональні вимоги можуть визначати максимальну кількість одночасних користувачів, час відповіді системи на запити або інші показники продуктивності.

Обмеження – це параметри, які обмежують розробників у використанні певних технологій або ресурсів. Обмеження можуть включати вимоги до бюджету, часу виконання, наявних людських ресурсів або існуючих систем, з якими новий продукт повинен бути інтегрований.

Критерії прийому – це правила та методи, за якими будуть оцінюватися результати роботи. Критерії прийому визначають, чи відповідає кінцевий продукт вимогам, зазначеним у ТЗ, і чи можна вважати проект завершеним. Наприклад, критерієм прийому може бути успішне проходження тестування, відповідність продукту функціональним вимогам або досягнення певних показників продуктивності.

План тестування та валідації – це розділ, який визначає методи перевірки роботи системи або продукту після його розробки. Тестування може включати різні види: функціональне тестування, навантажувальне тестування, тестування безпеки тощо. Валідація забезпечує впевненість, що кінцевий продукт відповідає вимогам замовника і правильно вирішує поставлені завдання.

Технічні завдання можуть відрізнятися за своєю структурою та змістом залежно від сфери бізнесу та типу продукту. Нижче наведені основні види ТЗ залежно від сфери діяльності:

ТЗ для IT-проектів: Цей вид ТЗ зазвичай містить опис програмного забезпечення або системи, які будуть розроблятися. ТЗ для IT-проектів часто містить UML-діаграми, схеми баз даних, алгоритми та специфікації API. Воно охоплює технічні аспекти роботи системи, функціональні та нефункціональні вимоги, а також опис взаємодії з іншими системами. Наприклад, для розробки ERP-системи можуть бути включені вимоги до інтеграції з іншими корпоративними системами.

ТЗ для мобільних додатків: Для мобільних додатків специфікація ТЗ акцентується на платформі (iOS, Android), адаптивності інтерфейсу під різні екрани, інтеграції з хмарними сервісами, функціональності офлайн-режиму та способах монетизації. Важливим аспектом є забезпечення інтуїтивної навігації та стабільної роботи навіть при обмежених ресурсах мобільного пристрою.

Часто використовуються макети екранів (wireframes), опис сценаріїв взаємодії з користувачем (user stories) і прототипи.

ТЗ для дизайну: У дизайні технічне завдання фокусується на візуальних та естетичних аспектах. ТЗ для дизайну може містити вимоги до стилю, кольорової палітри, типографіки, ілюстрацій, анімації тощо. Для друкованих матеріалів можуть бути вказані параметри для друкарського обладнання (формат, роздільна здатність, кольорова модель), а для веб- або мобільних продуктів — вимоги до адаптивності та респонсивності дизайну. Може включати moodboards або референси, які допоможуть зрозуміти візуальний стиль проекту.

ТЗ для вебсайтів: У технічному завданні для вебсайтів основний акцент робиться на архітектурі сайту, його структурі, навігації, інтеграції з базами даних та зовнішніми сервісами. ТЗ для вебсайтів може також містити вимоги до SEO-оптимізації, забезпечення безпеки, захисту від атак та швидкості завантаження сторінки. Наприклад, якщо це лендінг-пейдж для просування продукту, то важливо вказати вимоги до розміщення ключових елементів, таких як заклики до дії (CTA) та контактна форма.

ТЗ для промисловості: У технічному завданні для промислових проектів основним аспектом є технічні характеристики обладнання, вимоги до безпеки, стандарти якості та екологічні норми. У таких ТЗ часто включаються креслення, технічні схеми, інструкції з монтажу та експлуатації обладнання. Важливо враховувати міжнародні стандарти та норми, які регулюють виробництво, а також можливості подальшої автоматизації або модернізації систем.

ТЗ для рекламних кампаній: ТЗ для маркетингових проектів фокусується на креативній концепції, цільовій аудиторії, каналах просування та ключових показниках ефективності (KPI). Рекламні кампанії можуть включати вимоги до формату та стилю креативу, розміщення реклами на різних платформах (соціальні мережі, медіа, ТВ) та інші аспекти. ТЗ може включати план розміщення реклами, часові рамки та бюджет кампанії.

Технічне завдання має чітку структуру, яка дозволяє систематизувати інформацію та забезпечити зрозумілість для всіх учасників проекту. Основними елементами структури ТЗ є:

1. **Загальні відомості**: Назва проекту, замовник, виконавець, дати початку та завершення робіт, контактні особи.
2. **Опис проекту**: Загальна інформація про проект, мета, завдання, очікувані результати.
3. **Функціональні вимоги**: Опис основних функцій продукту, які повинні бути реалізовані в рамках проекту. Це може включати сценарії використання, діаграми або моделі.
4. **Технічні вимоги**: Опис технологій, які будуть використовуватися, вимоги до архітектури системи, баз даних, мови програмування, інтерфейсу користувача.

5. **Нефункціональні вимоги**: Вимоги до продуктивності, безпеки, надійності, масштабованості системи.
6. **Інтеграції**: Опис зовнішніх систем, з якими продукт буде взаємодіяти, вимоги до API та обміну даними.
7. **Тестування та валідація**: Опис методів перевірки працездатності системи, вимоги до тестування та критерії прийому роботи.
8. **План виконання**: Розподіл робіт на етапи, графік виконання, проміжні та кінцеві терміни.

4.5.1 Стандарти написання технічних завдань

Технічне завдання (ТЗ) вимагає певної структури та стандартів, щоб забезпечити чіткість і зрозумілість для всіх сторін, залучених до проекту. У різних країнах та галузях існують національні й міжнародні стандарти, які регулюють написання ТЗ, особливо у сфері розробки автоматизованих систем і програмного забезпечення.

Для забезпечення якості та стандартизації ТЗ у світовій практиці використовуються міжнародні стандарти, такі як ISO, IEEE, і ITIL. Вони встановлюють вимоги до структури, змісту та оформлення ТЗ, а також до взаємодії між сторонами, які беруть участь у проекті.

ISO (International Organization for Standardization) – це міжнародна організація, що розробляє стандарти для різних галузей, включно з інформаційними технологіями. Одним з основних стандартів, який регулює розробку програмного забезпечення та підготовку технічних завдань, є ISO/IEC/IEEE 29148:2018 **«Systems and software engineering – Life cycle processes — Requirements engineerin»**.

Основні аспекти цього стандарту включають:

Вимоги до структури ТЗ: вказується, що ТЗ повинно містити всі необхідні для проекту аспекти, включаючи вимоги до функціональності, безпеки, продуктивності, і т.д.

Методологія збору вимог: визначається, як правильно збирати та аналізувати вимоги від стейкхолдерів.

Валідація та верифікація вимог: ТЗ повинно бути написано таким чином, щоб забезпечити можливість верифікації (перевірки правильності формулювання вимог) і валідації (перевірки виконання вимог у кінцевому продукті).

Стандарт також розглядає поняття *системного інжинірингу*, де описується процес визначення і управління вимогами протягом життєвого циклу проекту, включно з початковими вимогами, змінами та доопрацюваннями.

IEEE 830-1998 (Institute of Electrical and Electronics Engineers) – це один із широко відомих стандартів, який визначає керівництво з написання *технічних вимог до програмного забезпечення (Software Requirements Specification, SRS)*.

Цей стандарт використовують для розробки ТЗ для проектів у галузі ІТ. Основні його пункти:

Мета: Вказує на важливість визначення чітких вимог до програмного забезпечення та структурування ТЗ у такий спосіб, щоб воно було зрозумілим як замовникам, так і розробникам.

Аспекти якості ТЗ: Стандарт вимагає, щоб вимоги в ТЗ були коректними, повними, однозначними, перевіреними, та не суперечливими.

Модульність: Важливий аспект структурування документа. ТЗ повинне бути модульним і мати зрозумілу ієрархію розділів.

Цей стандарт корисний для забезпечення послідовного підходу до збору вимог, розробки ТЗ і створення якісної документації для програмного забезпечення.

ITIL (Information Technology Infrastructure Library) – не є стандартом у прямому сенсі, але це бібліотека найкращих практик для управління ІТ-сервісами. В рамках ITIL технічне завдання має регулюватися процесами *Service Design і Service Transition*, де визначаються вимоги до створення і зміни ІТ-сервісів.

Стандарти ITIL вказують на необхідність детального опрацювання вимог у ТЗ, включаючи вимоги до:

- Безперервності сервісу.
- Захисту даних.
- Інтеграції з іншими системами.
- Керування доступом та правами користувачів.

Для складних ІТ-проектів ТЗ має відповідати вимогам з управління інфраструктурою ІТ.

В Україні також існують державні стандарти, які регулюють процес створення технічної документації, включно з технічними завданнями. Найбільш відомий стандарт для розробки автоматизованих систем – **ДСТУ 34.602-89**.

ДСТУ 34.602-89 (Система стандартів з автоматизованих систем) — цей стандарт встановлює загальні вимоги до створення технічного завдання на розробку автоматизованих систем управління, включно з інформаційними системами. Він передбачає створення ТЗ для всіх етапів життєвого циклу автоматизованих систем: від планування до експлуатації.

Основні вимоги стандарту:

- *Мета і завдання проекту*: в ТЗ повинна бути чітко сформульована мета створення системи та опис її основних функцій.
- *Опис середовища функціонування*: ТЗ повинне містити інформацію про середовище, в якому буде функціонувати система, включаючи технічні умови та обмеження.

- *Технічні вимоги*: вимагається детальне визначення технічних вимог, таких як обсяги пам’яті, продуктивність, безпека даних, надійність і т.д.
- *Опис етапів реалізації проекту*: передбачено етапний підхід до виконання робіт, включно з проміжними контрольними точками і термінами.

Цей стандарт є обов’язковим для державних проектів, але може бути використаний і в комерційних проектах, щоб забезпечити стандартизований підхід до розробки автоматизованих систем.

ДСТУ ISO/IEC 12207:2016 – цей стандарт є перекладом міжнародного стандарту ISO/IEC 12207 і регулює процеси життєвого циклу програмного забезпечення, включаючи вимоги до технічних завдань. Він детально описує процеси розробки, впровадження, тестування, підтримки та обслуговування програмного забезпечення. Основні положення стандарту:

- *Визначення ролей і обов'язків*: Стандарт визначає відповідальних осіб за розробку і виконання технічного завдання.
- *Управління змінами*: Описується, як правильно управляти змінами в вимогах на етапах реалізації проекту.
- *Валідація вимог*: ТЗ повинно бути перевірено і затверджено на відповідність технічним і бізнес-вимогам перед початком розробки.

Дотримання міжнародних і національних стандартів при написанні технічного завдання дозволяє:

- Забезпечити високу якість проектної документації.
- Полегшити комунікацію між замовником і виконавцем.
- Знизити ризики непорозумінь, конфліктів і необхідності переробки проекту.
- Забезпечити відповідність проекту вимогам безпеки та надійності.
- Полегшити управління проектом і його інтеграцію з іншими системами або проектами.

На прикладі реальних проєктів можна порівняти специфіку використання стандартів.

Для створення *ERP-системи для підприємства* часто використовують стандарти ISO і IEEE, які забезпечують чітке визначення вимог до інтеграції з іншими корпоративними системами (наприклад, бухгалтерія, управління персоналом), продуктивності та безпеки.

У проєктах із розробки *мобільного додатку* можуть бути застосовані стандарти ISO для визначення вимог до продуктивності додатку, його сумісності з різними платформами (iOS, Android) та вимог до безпеки користувацьких даних.

Розробка автоматизованої системи управління для державної установи. У цьому випадку використовується ДСТУ 34.602-89 для забезпечення

відповідності системи державним стандартам щодо захисту даних та інтеграції з іншими державними системами.

4.5.2 Приклади технічних завдань

Опираючись на вимоги в стандартах, що регламентують написання технічного завдання розглянемо кілька варіантів «типових» прикладів технічного завдання. Зокрема на розробку мобільного застосунку «Мій гаманець», який покликаний систематизувати персональні витрати та бюджет:

Проект: Мобільний застосунок "Мій гаманець"

1. Загальні відомості

- *Назва проекту: Розробка мобільного застосунку "Мій гаманець".*
- *Замовник: ТОВ "Фінансові рішення".*
- *Виконавець: Компанія "IT Solutions".*
- *Терміни розробки: 6 місяців.*
- *Контактні особи:*
  - *Від замовника: Іван Іванович, фінансовий директор, email: ivanov@fin.com*
  - *Від виконавця: Петро Петрович, проектний менеджер, email: petrov@itsolutions.com*

2. Мета проекту

Створити мобільний застосунок для управління фінансами користувачів, що дозволяє відстежувати стан рахунків, взаємодіяти з контрагентами та здійснювати фінансові операції.

3. Опис функціональності

3.1 Головна

- *Мета: Відображати актуальну інформацію про рахунки користувача.*
- *Функції:*
  - *Відображення балансу всіх рахунків (наприклад, готівкові кошти, банківські рахунки, електронні гаманці).*
  - *Перегляд детальної інформації по кожному рахунку.*
  - *Можливість додавання або редагування рахунків.*
  - *Графік руху коштів (діаграми з витратами та доходами).*

3.2 Довідники

- *Мета: Користувачі можуть керувати довідковими даними, які використовуються в транзакціях.*
- *Функції:*
  - *Валюта: Додавання нових валют, редагування курсів валют.*
  - *Контрагенти: Зберігання інформації про осіб чи організації, з якими здійснюються фінансові операції (ім'я, контактні дані).*
  - *Рахунки: Перелік усіх фінансових рахунків користувача (банківські рахунки, готівкові кошти, картки).*

- *Товари: Каталог товарів та послуг, які можуть бути використані в транзакціях (назва товару, категорія, ціна).*

3.3 Транзакції

- *Мета: Забезпечити користувачам можливість створювати нові транзакції для обліку доходів та витрат.*
- *Функції:*
  - *Додавання нових транзакцій (прибутки, витрати, перекази між рахунками).*
  - *Вибір рахунку, з якого списуються кошти.*
  - *Вибір контрагента і товару для транзакції.*
  - *Автоматичний розрахунок залишку на рахунку після транзакції.*
  - *Пошук транзакцій за датою, сумою, контрагентом.*

4. Технічні вимоги

- *Платформи: iOS (версія 12.0 і вище), Android (версія 8.0 і вище).*
- *База даних: SQLite для зберігання локальних даних користувача.*
- *Мова програмування: Swift для iOS, Kotlin для Android.*
- *Інтеграції: Інтеграція з API банківських сервісів для автоматичного оновлення даних про рахунки та транзакції.*
- *Безпека: Підтримка двофакторної аутентифікації (2FA) для захисту користувацьких акаунтів.*
- *Резервне копіювання: Автоматичне створення резервних копій даних на сервері для збереження історії транзакцій.*

5. Нефункціональні вимоги

- *Швидкодія: Час завантаження основного екрана — не більше 3 секунд.*
- *Надійність: Відновлення даних після збою повинно відбуватись автоматично.*
- *Інтерфейс: Простий і зручний у користуванні інтерфейс, що відповідає принципам UX/UI для фінансових застосунків.*
- *Масштабованість: Можливість додавання нових функцій (наприклад, підтримка криптовалют у майбутньому).*

6. Тестування

Методи тестування:

- *Тестування користувацького інтерфейсу.*
- *Тестування продуктивності (стабільність роботи при великій кількості транзакцій).*
- *Тестування безпеки (перевірка на захищеність від несанкціонованого доступу).*

Критерії прийняття:

- *Всі функціональні вимоги виконані.*
- *Інтерфейс працює без помилок на всіх підтримуваних платформах.*
- *Система відповідає вимогам продуктивності та безпеки.*

Це приклад детального технічного завдання для мобільного застосунку "Мій гаманець". Він охоплює всі основні аспекти, починаючи з загальної інформації про проект і закінчуючи технічними та нефункціональними вимогами. Така структура ТЗ дозволяє забезпечити якісне розуміння проекту розробниками, замовниками та тестувальниками.

Приклад технічного завдання на розробку сайту (Landing page) для просування мобільного застосунку «Мій гаманець»:

Проект: Розробка цільової сторінки для мобільного застосунку "Мій гаманець"

1. Загальні відомості

- *Назва проекту: Створення Landing page для мобільного застосунку "Мій гаманець".*
- *Замовник: ТОВ "Фінансові рішення".*
- *Виконавець: Веб-студія "Web Master".*
- *Терміни розробки: 2 місяці.*
- *Контактні особи:*
  - *Від замовника: Іван Іванович, маркетинг-директор, email: ivanov@fin.com*
  - *Від виконавця: Олександр Олександрович, веб-розробник, email: alex@webmaster.com*

2. Мета проекту

Створити привабливий та інформативний Landing page для презентації функцій мобільного застосунку "Мій гаманець" та стимулювання завантаження додатку через магазини додатків.

3. Опис функціональності

3.1 Головна сторінка

- *Мета: Привернути увагу відвідувачів та продемонструвати основні переваги застосунку.*
- *Функції:*
  - *Заголовок: Привітальний банер з назвою застосунку та його ключовою перевагою (наприклад, "Керуйте своїми фінансами легко!").*
  - *Кнопки завантаження: Кнопки для швидкого завантаження додатку з Google Play і App Store.*
  - *Опис продукту: Короткий опис основних функцій застосунку (управління рахунками, транзакціями, аналітика).*
  - *Промо-відео: Відеоролик, що демонструє роботу застосунку та його інтерфейс.*
  - *Соціальні докази: Відгуки від користувачів або рейтинги з магазинів додатків для підвищення довіри.*

3.2 Функціональні розділи

- *Мета: Надати відвідувачам детальну інформацію про функції застосунку.*
- *Функції:*

- *Фінансовий облік: Розділ з детальним описом функцій застосунку для управління рахунками (баланс, додавання рахунків).*
- *Транзакції: Опис можливостей управління транзакціями (додавання, редагування, пошук транзакцій).*
- *Довідники: Інформація про довідкові дані (валюти, контрагенти, товари), що користувач може редагувати.*
- *Аналітика: Розділ з графіками та діаграмами для візуалізації руху коштів.*

3.3 Кнопка заклику до дії (CTA)

- *Мета: Стимулювати відвідувачів завантажити застосунок.*
- *Функції:*
  - *Кнопка "Завантажити зараз" з посиланням на магазини додатків.*
  - *Відображення відгуків і кількості завантажень.*

3.4 Форма для збору контактів

- *Мета: Збір контактної інформації для подальшого маркетингу.*
- *Функції:*
  - *Поле для введення email-адреси.*
  - *Підписка на новини або акції.*
  - *Інтеграція з email-маркетинговими сервісами (Mailchimp, Sendinblue).*

3.5 FAQ

- *Мета: Відповіді на часті питання користувачів щодо застосунку.*
- *Функції:*
  - *Питання і відповіді на найпоширеніші запити (наприклад, "Як створити транзакцію?", "Як додати новий рахунок?").*

4. Технічні вимоги

- *Технології:*
  - *HTML5, CSS3, JavaScript для клієнтської частини.*
  - *PHP або Node.js для серверної частини.*
  - *Інтеграція з Google Analytics для збору статистики про відвідування сторінки.*
- *Адаптивний дизайн: Сайт повинен бути оптимізованим для всіх пристроїв (десктопи, планшети, мобільні телефони).*
- *SEO-оптимізація: Використання методів SEO для підвищення видимості сторінки у пошукових системах.*
- *Швидкість завантаження: Час завантаження сторінки — не більше 2 секунд.*
- *Кросбраузерність: Підтримка всіх основних браузерів (Chrome, Firefox, Safari, Edge).*

5. Нефункціональні вимоги

- *Надійність: Сайт повинен коректно працювати при великій кількості одночасних відвідувачів.*
- *Безпека: HTTPS-протокол для шифрування даних.*

- *Масштабованість: Можливість додавання нових функцій або розширення контенту (блоги, новини).*

6. Тестування

Методи тестування:

- *Тестування відображення на різних пристроях (адаптивність).*
- *Перевірка функціональності кнопок завантаження.*
- *Тестування форми збору контактів і інтеграції з email-маркетинговими сервісами.*

Критерії прийняття:

- *Сайт відповідає всім вимогам адаптивності та кросбраузерності.*
- *Інтерфейс працює коректно на всіх підтримуваних пристроях.*
- *Кнопки завантаження коректно перенаправляють на сторінки мобільних додатків у Google Play та App Store.*

Цей приклад технічного завдання для Landing page охоплює всі аспекти розробки маркетингової сторінки для просування мобільного застосунку. Чітка структура і продумані технічні та нефункціональні вимоги забезпечують успішне залучення нових користувачів.

4.6 Акти приймання

Факт виконання (поставки) постачальником послуг чи продуктів (результатів проєкту) замовнику фіксують актами чи сертифікатами. Таких сертифікатів (актів) може бути два види – Сертифікат попереднього прийняття (англ Preliminary Acceptance Certificate) та Сертифікат остаточного прийняття (англ. Final Acceptance Certificate).

FAC (Сертифікат остаточного прийняття) – це документ, який підтверджує, що проект або його частина були завершені і прийняті замовником. Це важливий крок у завершенні проекту, який свідчить про те, що всі вимоги були виконані.

PAC (Сертифікат попереднього прийняття) – це документ, який підтверджує, що частина проекту були завершені і прийняті замовником. Зазвичай, до таких сертифікатів прив'язують оплату за виконання певних об'ємів робіт чи поставок. В замовленні на придбання окремим пунктом описують порядок поставки и прийняття та графік платежів.

Для чого потрібен PAC/FAC:

Офіційне підтвердження завершення: FAC підтверджує, що проект виконано відповідно до вимог і специфікацій.

Підстава для оплати: Наявність FAC може бути необхідною умовою для остаточної оплати постачальнику або підряднику.

Закриття проекту: Використовується для документального підтвердження завершення проекту.

Розглянемо детальніше структуру PAC/FAC:

1. *Заголовок документа*: Назва документа (FAC), Номер сертифіката.
2. *Інформація про проект*: Назва проекту, Номер контракту, Дата видачі FAC.
3. *Опис виконаних робіт*: Детальний опис того, що було виконано в рамках проекту.
4. *Підтвердження відповідності*: Вказівка, що всі вимоги, умови та специфікації були виконані.
5. *Додаткова інформація*: Інформація про будь-які відкриті питання або недоліки (якщо такі є).
6. *Підписи*: Підписи уповноважених осіб з обох сторін (замовник, постачальник).

В окремих випадках такий сертифікат підписується із зауваженнями. Наприклад, в ході тестування були виявленні недоліки і замовник з постачальником домовились, що вони будуть усунені постачальником в певний термін але платіж буде здійснено вчасно. Це фіксується в даних сертифікатах (актах).

4.7 Протоколи

Всю важливу інформацію, що постійно надходить слід фіксувати в протоколах. Одним з таких протоколів є протокол зустрічі.

Протокол зустрічі (англ. Minutes of Meeting, MoM) – це документ, який фіксує всі основні моменти, обговорення, рішення та дії, які відбулися під час зустрічі або наради. MoM слугує офіційним записом, що допомагає учасникам зустрічі зберегти інформацію про обговорення, визначити наступні кроки та відповісти за завдання. Для чого потрібен MoM**:**

- *Документування*: MoM фіксує всі важливі рішення та обговорення, що дозволяє уникнути непорозумінь у майбутньому.
- *Комунікація*: Допомагає учасникам зустрічі та тим, хто не зміг бути присутнім, отримати зрозуміле уявлення про те, що було обговорено.
- *Контроль виконання*: MoM містить інформацію про призначені завдання, терміни виконання та відповідальних осіб, що допомагає відслідковувати прогрес.
- *Управління проектами*: У контексті управління проектами MoM є важливим інструментом для моніторингу рішень та дій, що впливають на хід проекту.

Структура MoM**:**

1. **Заголовок документа**:
   - Назва документа (Minutes of Meeting).
   - Дата та час зустрічі.
   - Місце проведення зустрічі.

2. **Список учасників**:

- Імена та посади всіх учасників зустрічі.

3. **Порядок денний**:

- Чіткий перелік тем, які були обговорені під час зустрічі.

2. Записи обговорень:

- Основні пункти обговорення для кожної теми, включаючи висловлені думки, коментарі та зауваження учасників.

3. Рішення:

- Фіксація всіх ухвалених рішень.

4. Дії та відповідальні особи:

- Перелік завдань, які потрібно виконати, разом із термінами та особами, відповідальними за їх виконання.

5. Дата наступної зустрічі (якщо застосовно):

- Інформація про заплановану наступну зустріч, якщо така є.

Розглянемо приклад Minutes of Meeting, оформлений у вигляді таблиці, (допустимо звичайним списком):

Назва: Щотищнева зустріч		
Дата: 30 вересня 2024	***Час:*** 10:00 - 11:00	***Місце:*** Конференц-зал 1
Учасники:		
1	Іван Петров	Керівник проекту
2	Олена Іванова	Бухгалтер
3	Сергій Коваленко	Технічний спеціаліст
Порядок денний:		
1	Обговорення бюджету проекту	
2	Огляд термінів виконання	
3	Розподіл завдань	
Обговорення:		
1	Бюджет проекту - Обговорено можливість скорочення витрат на 10%. *Олена* підготувала новий бюджет для наступної зустрічі.	
2	Терміни виконання - Визначено нові терміни завершення етапів проекту.	
3	Розподіл завдань - *Сергій* відповідає за завершення технічних специфікацій до *15 жовтня*.	
Наступна зустріч: 7 жовтня 2024, о 10:00.		

Такий протокол направляється всім учасникам зустрічі та, за замовчуванням, вважається вірним, якщо хтось із учасників не повідомив про зворотне.

4.8 NDA

Угода про нерозголошення (Non-Disclosure Agreement, NDA) – це юридичний документ, який укладається між двома або кількома сторонами з метою захисту конфіденційної інформації, що передається між сторонами. NDA визначає умови, за яких інформація може бути розкрита, а також обов'язки сторін щодо збереження конфіденційності.

Такий договір укладають компанії, коли починають співпрацю, наприклад на етапі участі в тендері. Також, такий договір може укладатися, між компанією і фізичною особою (виконавцем чи підрядником), перед ознайомленням з технічним завданням чи вимогами. Основні цілі NDA:

Захист конфіденційної інформації: NDA гарантує, що чутливі дані (наприклад, бізнес-плани, технології, фінансова інформація, клієнтські списки) не будуть розкриті третім особам.

Запобігання конкурентам: Використання NDA допомагає уникнути ситуацій, коли одна сторона може використати отриману інформацію на шкоду іншій стороні, наприклад, для конкуренції на ринку.

Правовий захист: NDA надає можливість подавати в суд у разі порушення умов конфіденційності, що служить додатковим стимулом для сторін дотримуватись угоди.

NDA також має визначену структуру:

Заголовок документа: Назва документа (Non-Disclosure Agreement).

Дані сторін: Імена та адреси сторін, що укладають угоду.

Визначення конфіденційної інформації: Чітке визначення, яка інформація вважається конфіденційною.

Зобов'язання сторін: Опис обов'язків сторін щодо збереження конфіденційності (наприклад, не розкривати інформацію третім особам).

Виключення: Інформація, яка не підлягає захисту (наприклад, інформація, яка вже є публічною або яку було отримано легально з інших джерел).

Термін дії: Вказівка на те, скільки часу угода буде чинною, і як довго інформація залишатиметься конфіденційною.

Права та обов'язки: Уточнення прав і обов'язків сторін у разі порушення умов NDA.

Підписи: Підписи уповноважених осіб від обох сторін.

5. ДИЗАЙН ІНФОРМАЦІЙНИХ СИСТЕМ

5.1 Основи дизайну в розробці інформаційних систем

Дизайн, як галузь існує приблизно 100 років. Він є результатом тисячолітньої історії розвитку людства і цивілізації. Вільям Морріс, художник і теоретик у галузі предметної творчості був лідером руху «За зв'язок мистецтв і ремесел» в Англії кінця XIX століття, що прийнято вважати початком розвитку дизайну.

У перших школах дизайну, які утворилися в 1920-х роках, поділ студентів був: столярна справа, гончарне ремесло, метало-обробка, ткацтво, графіка (поліграфія). Від самого початку класифікаційною ознакою, за якою поділяли дизайнерську діяльність, були матеріали, що з ними працював дизайнер.

Американський феномен виникнення дизайну пов’язують із всесвітньою кризою 1929 року, коли дизайн стає реальною комерційною силою, в результаті чого виникає професійна «індустрія дизайну». Сьогодні важко уявити яку-небудь сферу людської діяльності, де б не працював дизайнер. Сформувавшись у професійному середовищі архітекторів і художників, дизайн у процесі розвитку не тільки перетворився на самостійний вид проектно-художньої діяльності, а й сам почав активно впливати на художню форму, усе більше й більше розширюючи свої професійні сфери.

Із середини XX-го століття інформаційні системи набувають властивості візуального сприйняття, поступово відходячи від «чорного екрану», що якісно впливає на розвиток галузі. А також, поступово переходять на використання комп'ютера у своїй діяльності.

Сучасні комп'ютерні програми не тільки скорочують час роботи над проєктом, а й значно розширюють палітру графічних і технічних засобів дизайнера. Для цього сьогодні створені спеціальні пакети художньо-графічних і конструкторських програм, інженерних, настільних видавничих систем.

Однією з найстаріших та найпоширеніших на сьогоднішній день ділянок дизайнерської діяльності є графічний дизайн (оформлення книг, рекламно-інформаційних проспектів, буклетів, плакатів, упаковок, етикеток, торговельних марок, фірмових знаків, шрифтових гарнітур, заставки й рекламні ролики на телебаченні тощо).

Дизайн (від лат. designare – вказувати, креслити, англ. design – задум, план, проект) – це процес планування та створення об'єктів, інтерфейсів, систем або досвідів, орієнтованих на вирішення конкретних проблем або потреб користувачів.

Він охоплює як естетичну складову, так і функціональну, забезпечуючи гармонійне поєднання форми та змісту. *Дизайнери* працюють над тим, щоб створити продукти або рішення, які є як привабливими, так і зручними для користувача. Основні аспекти (принципи) дизайну:

Функціональність: продукт повинен виконувати свої завдання ефективно.

Естетика: продукт повинен бути візуально привабливим.

Зручність використання: користувач повинен легко розуміти, як працює продукт.

5.2 Напрямки в дизайні

Розвиток галузі досяг небачених масштабів, чим спричинив її суттєвий поділ на напрямки (підгалузі). Розглянемо кілька популярних підгалузей дизайну:

Графічний дизайн – це творча дисципліна, яка поєднує візуальне мистецтво та комунікацію для передачі ідей, повідомлень чи інформації через зображення, текст і символи (рис. 5.1). Основна мета графічного дизайну — створювати візуально привабливі та функціональні роботи, які ефективно доносять необхідну інформацію до аудиторії.

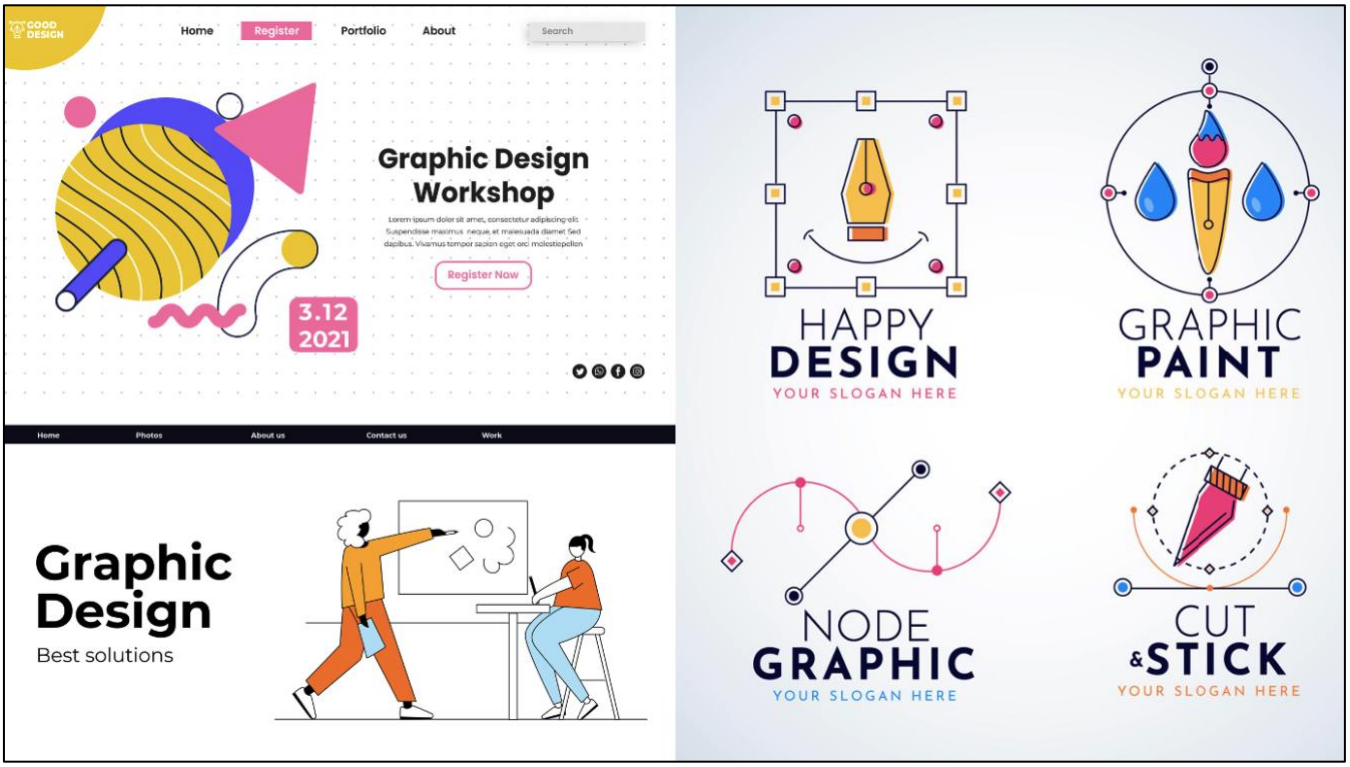

Рисунок 5.1 – Графічний дизайн.

Графічний дизайн відіграє вирішальну роль у формуванні брендів, рекламних кампаній і комунікаційних стратегій. Він допомагає створювати візуальну ідентичність, яка виділяє продукт або компанію на ринку, залучає споживачів і сприяє ефективному донесенню повідомлень.

UX/UI дизайн – це ключові аспекти розробки користувацьких інтерфейсів для веб-сайтів, мобільних додатків та інших цифрових продуктів (рис. 5.2). Він охоплює два взаємопов'язані напрями: *UX (User Experience, з англ. Досвід Користувача)*, що фокусується на досвіді користувача, і *UI (User Interface)*, що зосереджений на візуальному та інтерактивному дизайні інтерфейсу.

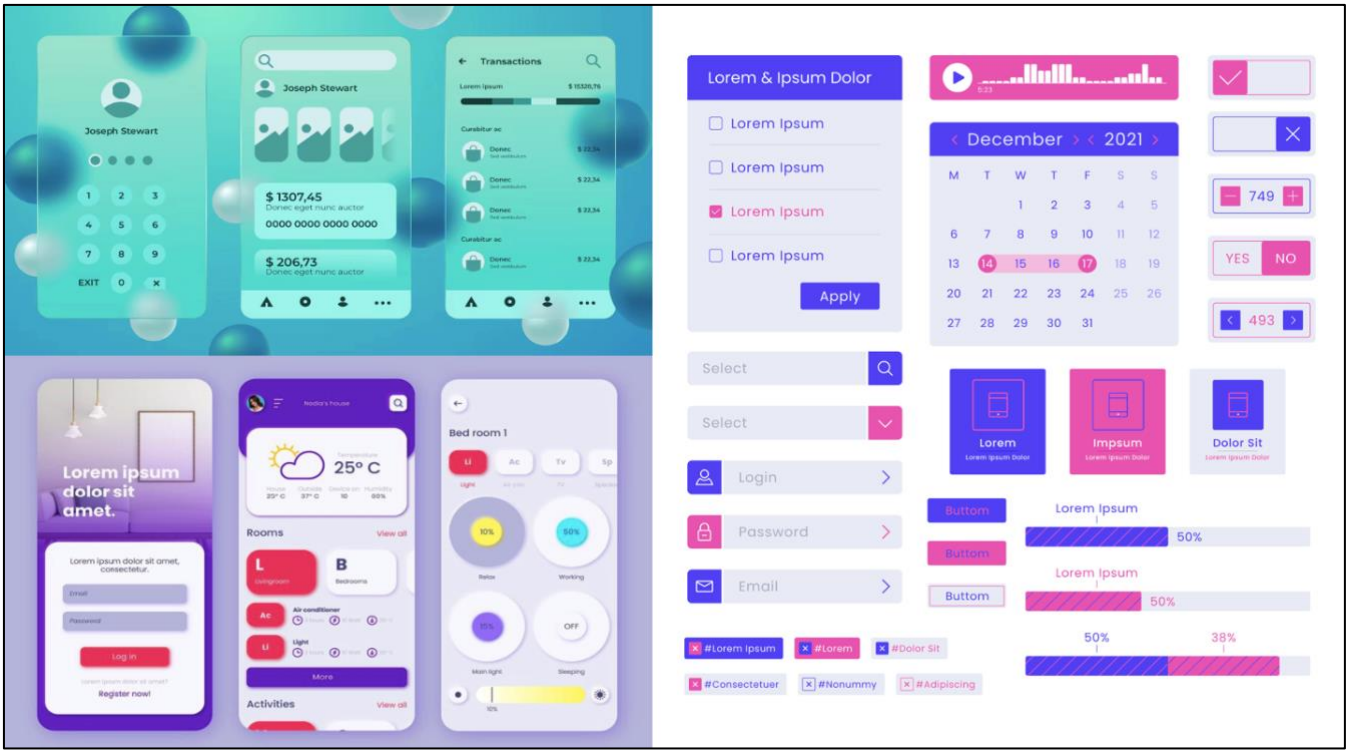

Рисунок 5.2 – Приклади UX/UI дизайну.

UX-дизайн стосується загального досвіду користувача при взаємодії з продуктом. Основна мета – зробити користування зручним, інтуїтивним і приємним, а також вирішити проблеми користувачів ефективно.

UI-дизайн стосується зовнішнього вигляду та відчуття інтерфейсу користувача. Він охоплює естетичні та функціональні аспекти інтерфейсу, включаючи кольорову палітру, типографіку, розташування елементів і загальну візуальну гармонію. UX визначає, *як продукт працює*, а UI відповідає за те, *як він виглядає*. UX-дизайн фокусується на оптимізації шляхів користувача і вирішенні проблем, тоді як UI-дизайн створює візуальне оформлення і забезпечує інтуїтивну взаємодію з користувачем.

Промисловий дизайн – це галузь дизайну, яка займається розробкою зовнішнього вигляду, функціональності та ергономіки масових продуктів промислового виробництва (рис. 5.3). Промислові дизайнери працюють над тим, щоб поєднати естетику з функціональністю, зручністю використання і технологічними можливостями виробництва, враховуючи економічні та екологічні фактори.

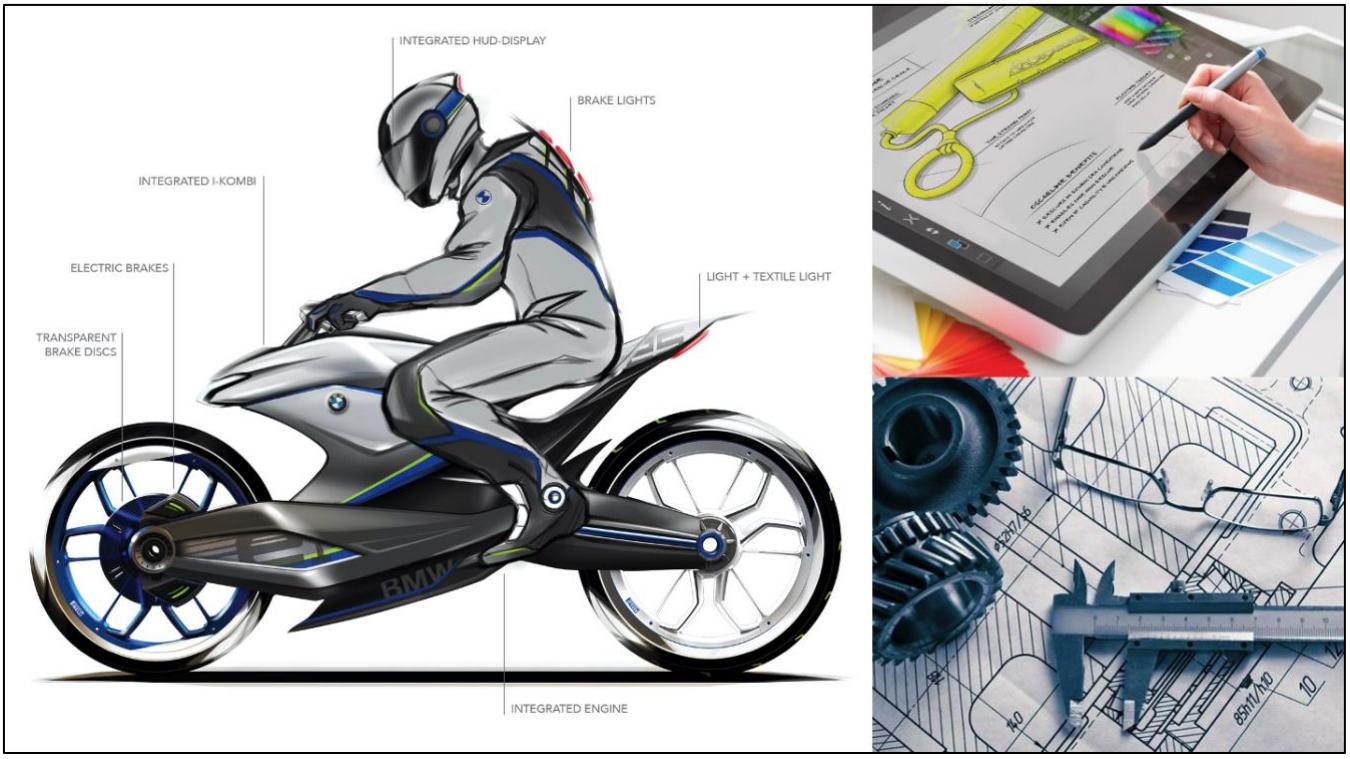

Рисунок 5.3 – Приклади промислового дизайну.

Промисловому дизайну притаманна характеристика – ергономіка. Вона стосується комфорту і зручності використання продукту. Промисловий дизайнер враховує людську анатомію та фізіологічні особливості, щоб створити продукти, які легко використовувати без дискомфорту. Сучасний промисловий дизайн все частіше враховує екологічні аспекти, такі як використання екологічно чистих матеріалів, енергоефективність і можливість вторинної переробки продукту після закінчення терміну його служби.

Промисловий дизайн застосовується в галузях розробки побутової техніки (дизайн побутових приладів), розробка транспортних засобів (автомобілі, поїзди, велосипеди та іншими засоби транспорту), електроніка (смартфони, ноутбуки, навушники), меблі (створення ергономічних, стильних і практичних виробів), медичні пристрої, тощо.

Промисловий дизайн має значний вплив на те, як люди взаємодіють із продуктами, що їх оточують у повсякденному житті. Він поєднує інженерні аспекти з креативністю, що дозволяє створювати продукти, які задовольняють потреби користувачів і є конкурентоспроможними на ринку. Від хорошо спроектованих промислових продуктів залежить не лише зручність користувача, а й економічний успіх компаній, які їх виробляють.

Архітектурний дизайн – це процес створення будівель, споруд і середовища для життя та роботи, поєднуючи естетичні, функціональні, технічні й екологічні аспекти. Архітектори займаються не лише плануванням зовнішнього вигляду будівель, але й розробкою їхньої внутрішньої структури, орієнтованої на комфорт та безпеку людей (рис. 5.4).

Рисунок 5.4 – Приклади архітектурного дизайну.

Архітектурний дизайн визначає, як будівля чи простір будуть використовуватись. Архітектори проектують споруди таким чином, щоб вони відповідали конкретним функціям – чи то житлові будинки, чи офіси, чи громадські будівлі. Важливо створити планування, яке відповідає вимогам користувачів і забезпечує ефективне використання простору.

Зовнішній вигляд будівель має значення для їхньої привабливості й відповідності навколишньому середовищу. Архітектурний дизайн поєднує гармонію, пропорції, стиль та матеріали для створення естетично приємних споруд. Це може включати класичні стилі, модернізм, хай-тек чи інші архітектурні напрями.

Архітектори працюють над забезпеченням безпечності та надійності будівлі, враховуючи інженерні аспекти. Це включає вибір матеріалів, структурних елементів (колони, балки), методи будівництва та використання сучасних технологій, щоб споруди могли витримувати навантаження і вплив зовнішніх факторів (вітер, землетруси, сніг).

Сучасний архітектурний дизайн враховує екологічні фактори: енергоефективність, використання природних матеріалів, інтеграцію відновлюваних джерел енергії (сонячні панелі, системи для збирання дощової води) та мінімізацію впливу на довкілля. Архітектори також розробляють проекти, які сприяють збереженню ресурсів та підвищують якість життя в умовах урбанізації.

Архітектурний дизайн враховує потреби людей, які будуть використовувати простір. Це стосується як розташування функціональних зон (кухня, вітальня, ванні кімнати), так і освітлення, вентиляції, звукоізоляції та зручності

використання приміщень. Добре спроектований простір має відповідати фізіологічним та психологічним потребам людей.

Архітектори мають враховувати не лише дизайн окремої будівлі, а й те, як вона вписується у навколишнє середовище та міську інфраструктуру. Це стосується взаємодії з іншими будівлями, транспортними шляхами, публічними просторами та природними елементами (парки, водойми). Гармонійне інтегрування будівлі у міський контекст є важливим аспектом архітектурного дизайну.

Архітектурний дизайн має безліч різновидів: *житлова архітектура* (проектування житлових будинків, багатоквартирних будинків, котеджів та інших житлових приміщень), *громадська архітектура* (проектування шкіл, лікарень театрів, музеїв, офісних центрів, торговельних комплексів), *промислова архітектура* **(**заводи, фабрики, склади та інших об'єктів промислового значення), *ландшафтна архітектура* **(**проектування зовнішніх просторів, таких як парки, сквери, сади та публічні зони).

Архітектурний дизайн має велике значення для якості життя людей. Він формує не лише зовнішній вигляд міського середовища, а й визначає те, наскільки комфортними, функціональними і безпечними будуть будівлі та приміщення, якими користуються люди щодня. Архітектори відіграють важливу роль у створенні простору, що відповідає як практичним потребам, так і естетичним вимогам суспільства.

Отже, дизайн – це комплексний процес, який поєднує креативність з технічними навичками, і його мета – створення ефективних та естетично привабливих рішень для різних сфер діяльності.

5.3 Концепція дизайну інформаційних систем та програмних продуктів

В розробці сучасних програмних комплексів та інформаційних систем дизайн це «те як система працює, а не тільки як вона виглядає». Тобто про візуальну складову інформаційної системи говорять як про можливість взаємодіяти із нею. Так з'являється поняття інтерфейс (англ. interface). Розглянемо основні поняття в дизайні, що є фундаментальними у створенні ефективного та привабливого дизайну (інтерфейсу), незалежно від його сфери застосування.

1. *Композиція* є ключовим поняттям у дизайні, що стосується організації елементів на сторінці або в просторі (рис. 5.5). Вона охоплює розташування, співвідношення розмірів, пропорції та баланс елементів. Добре організована композиція допомагає створити гармонію та робить дизайн візуально привабливим і функціональним.

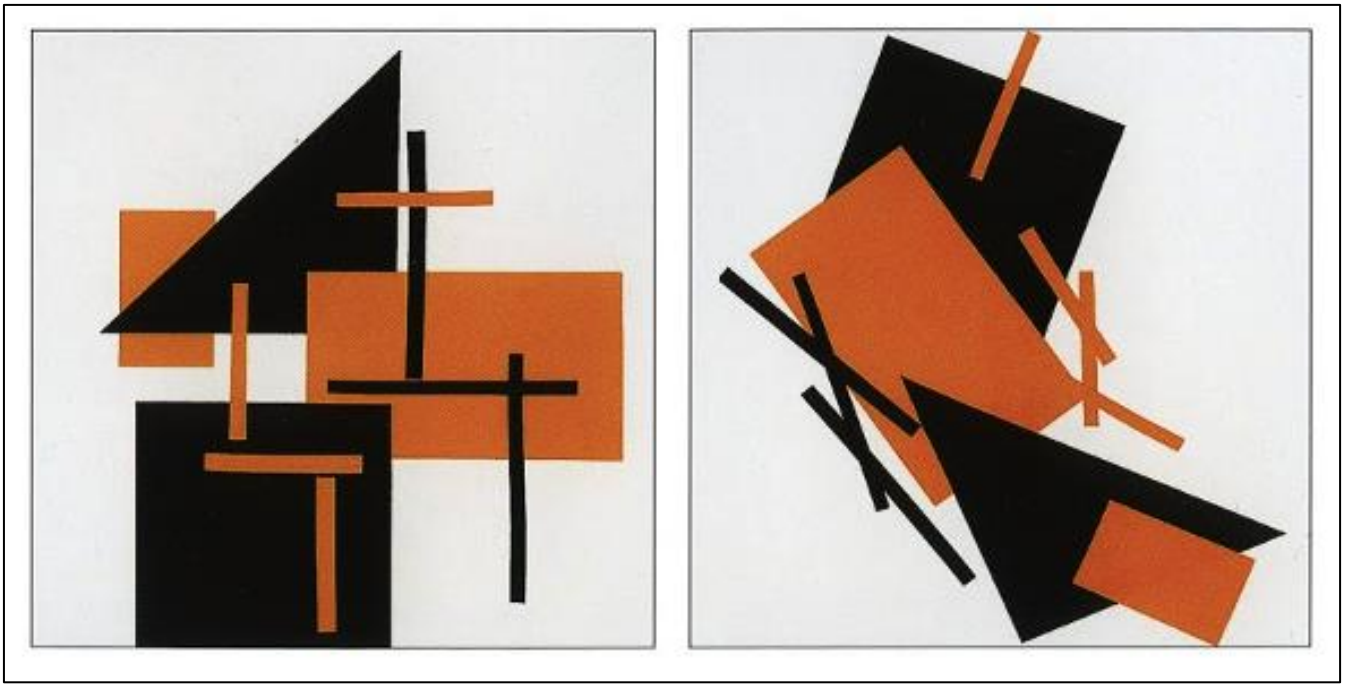

Рисунок 5.5 – Приклади композиції в дизайні.

2. *Контраст* використовується для підкреслення відмінностей між елементами дизайну (рис. 5.6). Це може бути різниця в кольорі, розмірі, формі або текстурі. Контраст допомагає привертати увагу до ключових частин дизайну та полегшує сприйняття інформації.

Рисунок 5.6 – Приклади контрасту в дизайні.

3. *Баланс* – це рівномірний розподіл візуальної ваги елементів у дизайні (рис. 5.7). Баланс може бути симетричним (елементи рівномірно розташовані з обох сторін) або асиметричним (різні елементи врівноважують один одного за допомогою розміру, кольору або розташування).

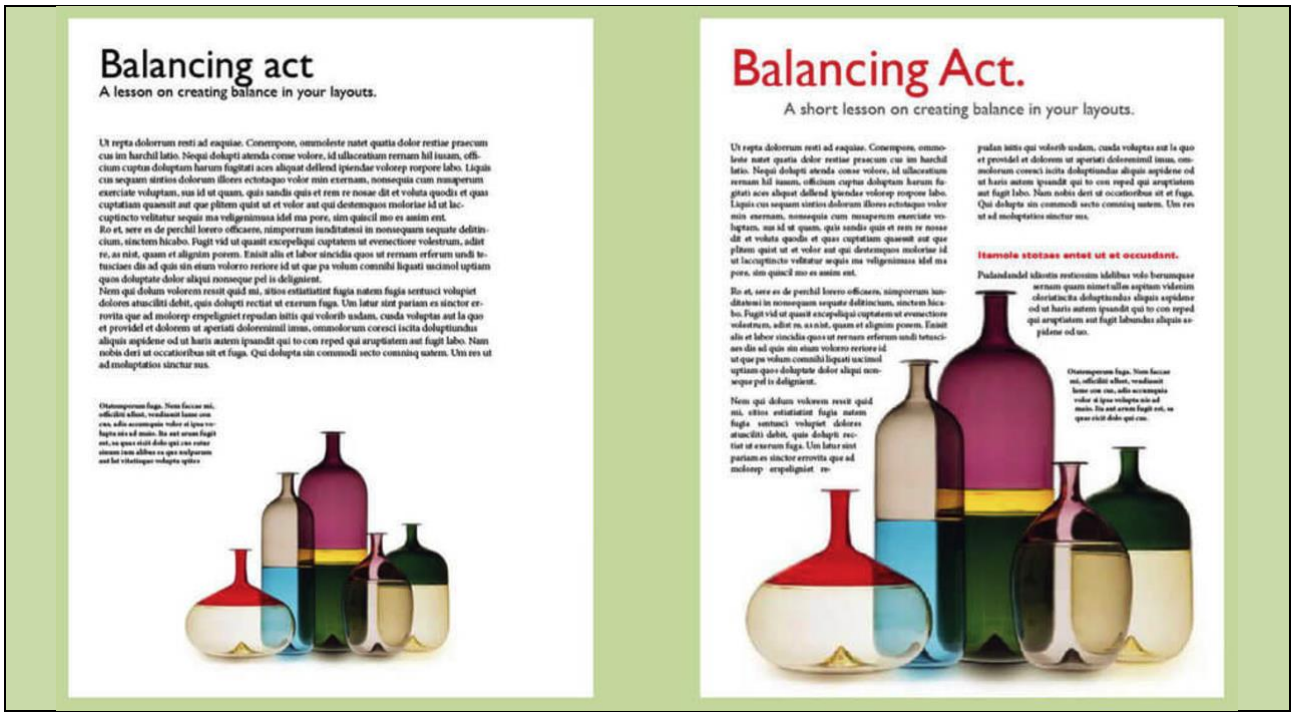

Рисунок 5.7 – Приклади балансу в дизайні.

4. *Пропорції* – це співвідношення розмірів елементів дизайну (рис. 5.8, 5.9). Правильне використання пропорцій допомагає створювати гармонійний і естетичний вигляд. Зокрема, часто використовується принцип «золотого перетину» для визначення ідеальних пропорцій.

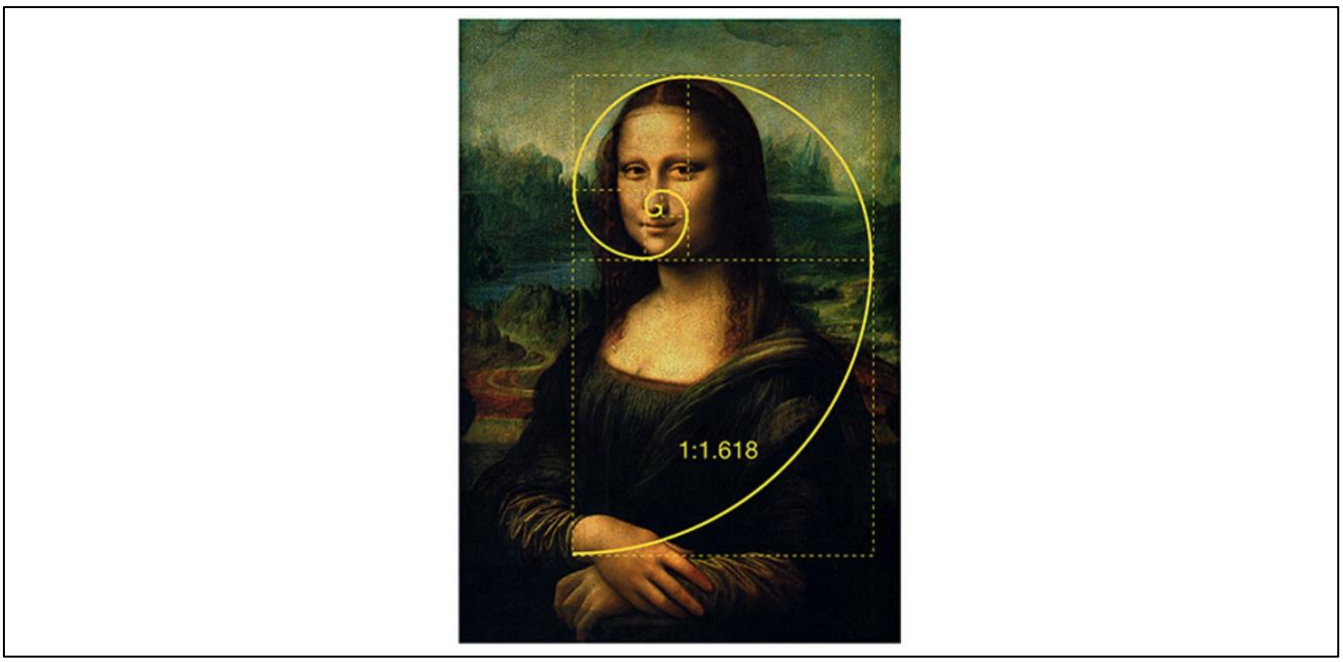

Рисунок 5.8 – Приклади пропорції в дизайні.

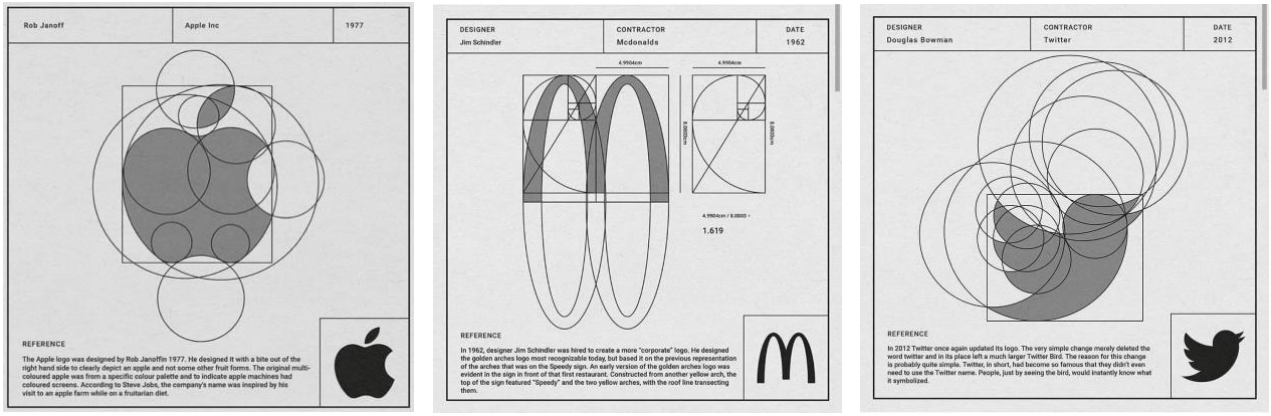

Рисунок 5.9 – Приклади використання золотого перетину в дизайні.

5. *Типографіка* стосується вибору шрифтів, їх розмірів, інтервалів між літерами, словами та рядками (рис. 5.10). Типографіка грає важливу роль у візуальній комунікації, оскільки допомагає підкреслити важливість тексту та полегшити його читання.

Рисунок 5.10 – Приклади типографіки в дизайні.

6. *Кольорова палітра* – це набір кольорів, які використовуються в дизайні для створення емоційного впливу (рис. 5.11). Правильний вибір кольорів впливає на сприйняття дизайну користувачем, створює атмосферу і настрій.

Рисунок 5.11 – Приклади кольорових палітр в дизайні.

Дизайнер створює низку документів, які допомагають організувати процес розробки, відобразити ідеї та передати їх іншим членам команди або клієнтам. Ось основні види документів, які створюють дизайнери:

Мудборд (англ. Moodboard) – це візуальний інструмент, який допомагає дизайнеру представити атмосферу, кольорову гаму, шрифти та загальний стиль майбутнього дизайну (рис. 5.12). Це колекція зображень, текстур і прикладів, що формують загальний настрій проекту.

Рисунок 5.12 – Приклади мудборду.

Ескізи (англ. Wireframes) — це чорно-білі схеми, що показують структуру і розташування елементів на сторінках або екранах без детального опрацювання дизайну (рис. 5.13). Вони використовуються на ранніх етапах для узгодження загальної концепції з клієнтом і розробниками.

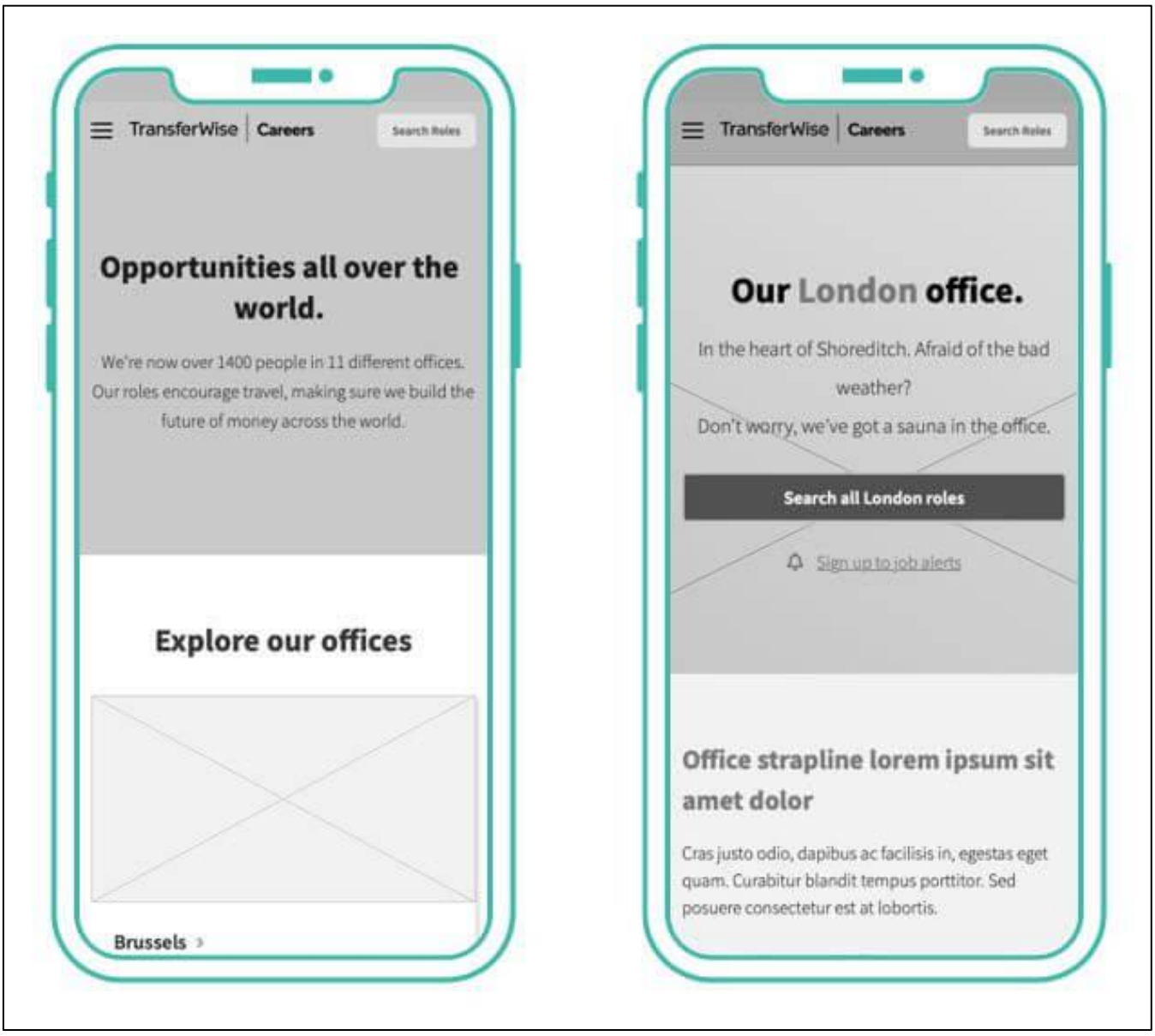

Рисунок 5.13 – Приклади ескізів.

Макет (англ. Mockup) – це більш деталізований прототип дизайну, що демонструє, як буде виглядати кінцевий продукт (рис. 5.14). У макеті показуються кольори, шрифти, зображення та інші графічні елементи. Це статичне представлення готового дизайну, яке часто використовується для узгодження з клієнтом.

Рисунок 5.14 – Приклади макетів.

Прототип – це інтерактивний макет, який дозволяє тестувати функціональність інтерфейсу до його розробки (рис. 5.15). Використовується в UX/UI дизайні для демонстрації та перевірки користувацького досвіду.

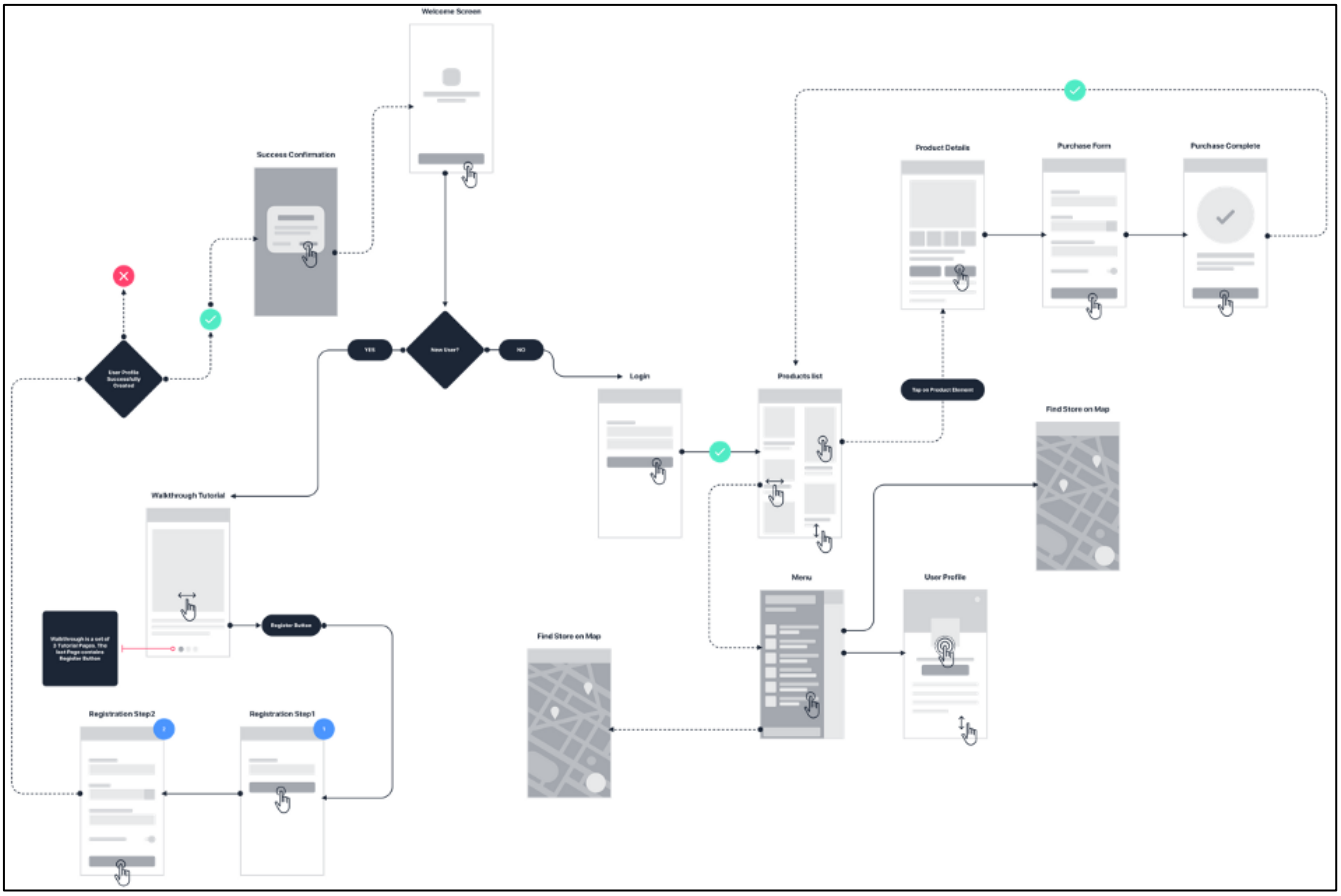

Рисунок 5.15 – Приклади прототипу.

Дизайн-система – це набір правил, стандартів і шаблонів, що використовуються для створення узгодженого дизайну в рамках великого проекту (рис. 5.16). Вона містить опис кольорів, шрифтів, стилів елементів інтерфейсу та інші компоненти.

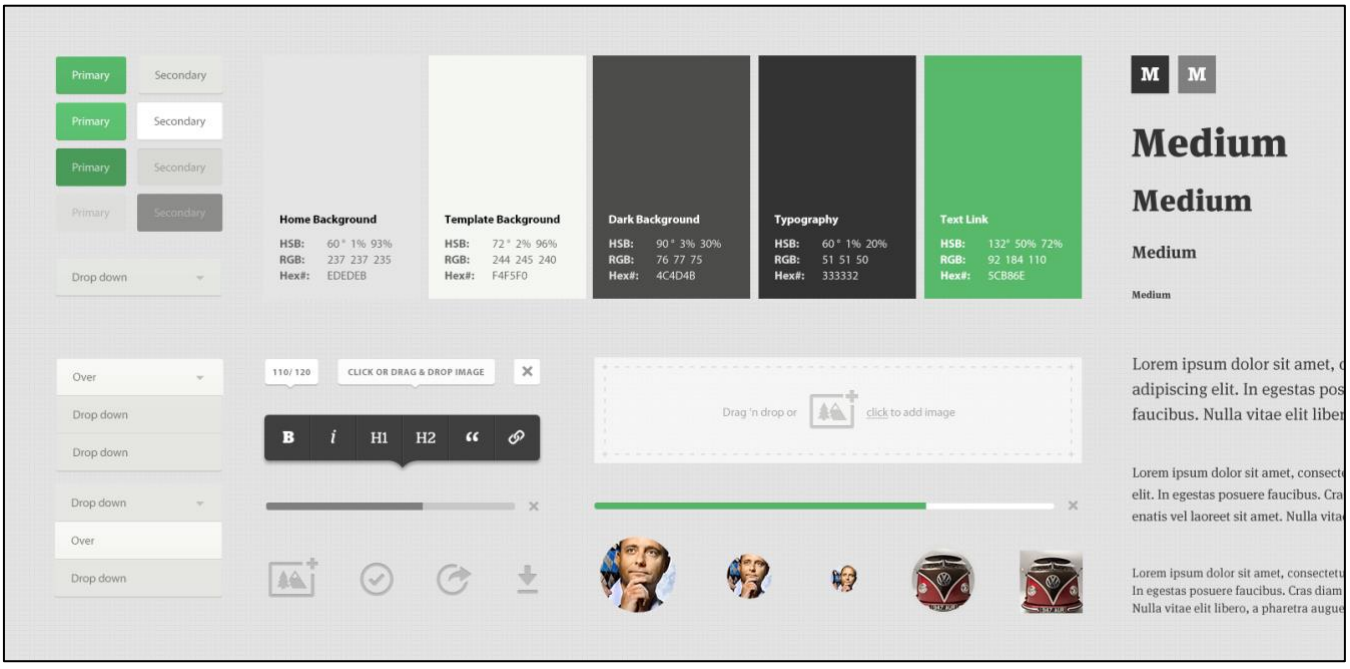

Рисунок 5.16 – Приклади дизайн-системи.

Технічна специфікація (англ. Design Specs) – це документ, який містить детальні технічні інструкції для розробників про те, як має бути реалізований дизайн. Специфікація включає розміри елементів, кольорову палітру, шрифти, відступи та інші параметри.

Стильовий гайд (англ. Style Guide) – це документ, який регламентує всі аспекти візуального стилю проекту: кольори, шрифти, розміри елементів, іконки, відступи та інші естетичні елементи. Стильовий гайд забезпечує єдиний вигляд продукту на всіх платформах і під час всіх етапів розробки.

Аудит інтерфейсу (англ. UI Audit) – це детальний аналіз існуючого інтерфейсу продукту. У ньому перевіряється відповідність стандартам дизайну, виявляються проблеми з узгодженістю елементів і пропонуються покращення для більш ефективної взаємодії з користувачем.

Флоучарти (англ. Flowcharts) – це схеми, що показують послідовність дій користувача в процесі виконання завдань у додатку чи на веб-сайті. Вони допомагають зрозуміти, як саме користувачі переміщуються між різними екранами або сторінками, і є корисними для планування логіки інтерфейсу.

Брендбук (англ. Brand Book або Brand Guidelines або Brand Manual чи Brand Style Guide) – це документ, який містить візуальні та комунікаційні стандарти бренду. Він орієнтований на широкий спектр використання і охоплює як цифрові, так і друковані матеріали. Брендбук зазвичай включає рекомендації не лише щодо дизайну, але й щодо тональності комунікації з аудиторією.

Дизайн система і Брендбук дуже схожі, на перший погляд, проте ці документи мають суттєві відмінності, хоча і стосуються візуальних аспектів компанії чи продукту але мають деякі спільні елементи.

Основні характеристики дизайн-системи:

- *Компоненти інтерфейсу*: включає бібліотеки інтерфейсних елементів (кнопки, форми, іконки, навігаційні елементи), які використовуються в додатках або на сайтах.
- *Функціональні стандарти*: визначає, як ці компоненти взаємодіють, як вони поводяться при різних сценаріях використання.
- *Адаптивність*: забезпечує єдність у дизайні на різних платформах і пристроях (мобільний, десктоп, планшет).
- *Комунікація між командами*: полегшує роботу між дизайнерами та розробниками, надаючи чіткі інструкції для реалізації інтерфейсних елементів.

Дизайн-система спрямована на зручне та послідовне створення цифрових продуктів і має високий рівень деталізації технічних аспектів, що стосуються інтерфейсу користувача.

Основні характеристики брендбука:

- *Логотип і його варіації*: містить правила використання логотипу в різних ситуаціях (фон, розмір, колір, варіанти розташування).
- *Кольорова палітра бренду*: визначає основні та додаткові кольори, які асоціюються з брендом.
- *Типографіка*: визначає шрифти, які використовуються в комунікації бренду.
- *Візуальні елементи*: фірмові патерни, ілюстрації, стилі фотографій або графіки, що використовуються для послідовності візуальної комунікації.
- *Тон голосу*: описує, як бренд "говорить" зі своєю аудиторією — офіційно чи неформально, емоційно чи нейтрально.
- *Застосування на різних носіях*: правила використання візуальних елементів у друкованих матеріалах, рекламі, упаковці, зовнішній рекламі, цифрових медіа і т.д.

Таблиця 5.1 – Основні відмінності дизайн-системи і брендбуку

	Дизайн-система	**Брендбук**
Призначення	більше орієнтована на створення цифрових інтерфейсів і забезпечує консистентність компонентів у продукті (вебсайт, додаток).	охоплює ширший спектр застосувань і стосується візуальної ідентичності бренду в цілому, від логотипу до маркетингових матеріалів.
Сфера застосування	застосовується в цифрових продуктах (UX/UI), а її основними користувачами є дизайнери та розробники.	використовується для всіх типів медіа (друкованих, цифрових), охоплюючи всі аспекти візуальної комунікації бренду.
Технічний рівень	має технічну структуру з детальними інструкціями для реалізації елементів інтерфейсу.	більш високорівневий документ, що описує загальні правила ідентичності бренду.

Отже, дизайн-система і брендбук доповнюють один одного, але вони вирішують різні завдання: перша забезпечує послідовність цифрових продуктів, а друга формує загальну ідентичність бренду у всіх середовищах.

6. РОЗРОБКА ПРОГРАМНИХ ПРОДУКТІВ

6.1 WEB-технології розробки

HTML (HyperTextMarkupLanguage – мова розмітки гіпертексту) – це мова тегів, засобами якої здійсюється розмічання веб-сторінок для мережі Інтернет. Веб-браузери отримують HTML-документи з веб-сервера або з локальної пам'яті й передають документи в мультимедійні веб- сторінки. HTML описує структуру веб-сторінки семантично і початкові включені сигнали для зовнішнього вигляду документа.

HTML дозволяє визначити структуру електронного документа з поліграфічним рівнем оформлення. Підсумковий документ може містити різноманітні елементи: ілюстрації, аудіо- і відеофрагменти. Мова HTML містить розвинені засоби для визначення кількох рівнів заголовків, шрифтових виділень, різних груп об'єктів та багато інших можливостей.

Гіпертекстова база даних у концепції WWW – це набір текстових файлів, розмічених мовою HTML, яка визначає форму подання інформації (розмітка) і структуру зв'язків цих файлів (гіпертекстові посилання). Основні частини HTML-елемента зображені на рис. 6.1.

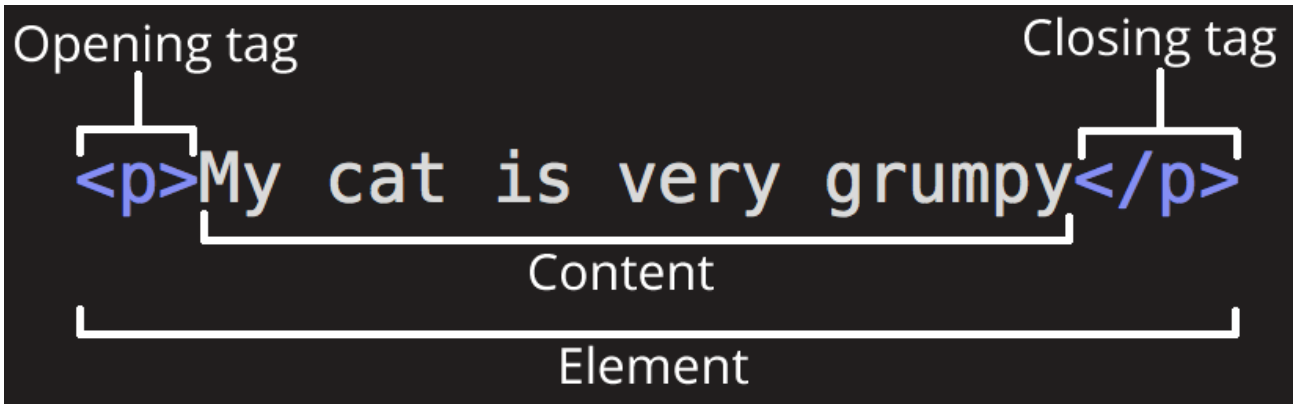

Рисунок 6.1 – Анатомія елемента HTML.

Початковий тег: містить назву елемента (в даному випадку), загорнену в кутові дужки. Цей тег позначає початок елемента, або місце, де елемент починає діяти. У даному випадку це місце, де починається параграф.

Кінцевий тег: такий самий тег, як і початковий, тільки тепер він містить косу риску перед назвою елемента. Цей тег позначає місце закінчення елемента; в даному випадку місце, де закінчується параграф. Одна з найпоширеніших помилок початківців – це забути поставити кінцевий тег, що може призвести до несподіваних результатів.

Вміст: вміст елемента, у даному випадку – просто текст.

Елемент: початковий тег плюс кінцевий тег плюс вміст між ними – це елемент.

Елементи також можуть мати атрибути (рис. 6.2). Атрибути містять додаткову інформацію про елемент, яка не відображається в самому контенті.

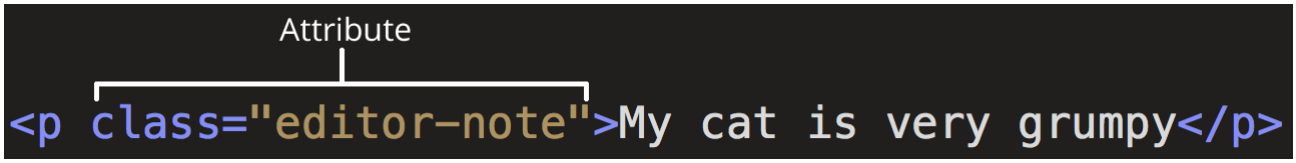

Рисунок 6.2 – Атрибут елемента HTML.

Як бачимо *class* – це ім'я атрибута, а *editor-note* – це значення атрибута. Атрибут class дозволяє дати елементу певний ідентифікатор, який пізніше можна використати для того, щоб «доступитися» (достукатись) до цього елемента і змінити його стиль.

Елементи HTML є будівельними блоками веб-сторінок. За допомогою конструкцій HTML зображення та такі інші об'єкти, як інтерактивні форми, можуть бути вбудовані у візуалізовану сторінку. HTML надає засоби для створення структурованих документів, позначаючи структурну семантику тексту, наприклад заголовки, абзаци, списки, посилання, цитати та інші елементи.

Браузери не показують теги HTML, але використовують їх для інтерпретації вмісту сторінки.

Каскадні таблиці стилів(CSS) – це мова стилів, яка використовується для опису подання документа, написаного HTML чи XML (охоплюючи такі XML-діалекти, як SVG або XHTML). CSS описує як елементи мають відображатися на екрані, на папері, при озвученні, або на інших пристроях.

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

CSS простими словами можна охарактеризувати як набір правил, що описують, як має виглядати елемент. Правило складається з селектора і блоку оголошень (див. рис. 6.3).

Рисунок 6.3 – Структура мови CSS.

Селектор повідомляє, до якого елемента будуть застосовані описувані в CSS властивості стилю. В якості селектора може виступати будь-який тег, якому задається форматування (розмір, колір і т. д.).

Блок оголошень складається з пар «властивість: значення» (записи завжди чергуються двокрапкою), розміщених в фігурних дужках. Записи закінчуються крапкою з комою. CSS нечутливий до табуляції, пропусків, регістра.

Скрипт (script) – це поняття в програмуванні, що позначає послідовність команд для виконання конкретних операцій. По суті, це невелика програма, яка відповідає певній дії.

Скрипти використовуються у веб-розробці, зокрема, щоб автоматизувати операції. Існують скриптові мови програмування, які називають «мовами сценаріїв». Сценарії – друга назва скриптів, і вони поділяються на дві категорії:

- *клієнтські*, які виконуються на призначеному для користувача ПК (команди в коді web-ресурсу);
- *серверні* – для виконання різних функцій в структурі ресурсу.

Сценарії пишуть за допомогою спеціальних мов, які називають скриптовими. У них різний синтаксис, можливості і сфери застосування. Мови бувають трьох видів:

- *командно-сценарні* – призначені для виконання дій в операційних системах; робота в консолі операційної системи проводиться на командно-сценарному рівні мовою цієї системи (PowerShell, Bash і т.д.);
- *вбудовані (прикладні)* – використовуються під конкретні завдання і зазвичай є внутрішніми мовами будь-якої системи. Наприклад, AutoLISP застосовується в САПР AutoCAD;
- *загального призначення* – підходять для широкого спектра завдань. До них належить JS і більшість інших мов програмування, що застосовуються в веб-розробці.

Типами **баз даних**, які називаються також **моделями БД або сімействами БД**, є шаблони і структури, які використовуються для організації даних в *системі управління базами даних* (СУБД). Вибір типу вплине на те, які операції зможе виконувати додаток та як будуть представлені дані. Найпростіший спосіб зберігання даних – текстові файли (рис. 6.4).

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
bob:x:1000:1000:Bob Smith,,,:/home/bob:/bin/bash
```

Рисунок 6.4 – Текстовий тип БД (найпростіший).

Метод застосовується і сьогодні для роботи з невеликими обсягами інформації. Для поділу полів використовується спеціальний символ: кома або крапка з комою в csv-файлах датасета, двокрапка або пробіл в * nix- подібних системах.

На відміну від текстових таблиць, в ієрархічних базах даних кожен запис має одного «батька». Це створює деревоподібну структуру, в якій записи класифікуються за їхніми стосунками з ланцюжком батьківських записів. Приклади: файлові системи, DNS, LDAP.

Реляційні бази даних (рис. 6.5) – найстаріший тип досі широко використовуваних БД загального призначення. Дані та зв'язки між даними організовані за допомогою таблиць. Кожен стовпець у таблиці має ім'я і тип. Кожен рядок є окремим записом або елементом даних в таблиці, який містить значення для кожного зі стовпців. Приклади:MySQL, MariaDB, PostgreSQL, SQLite. 50

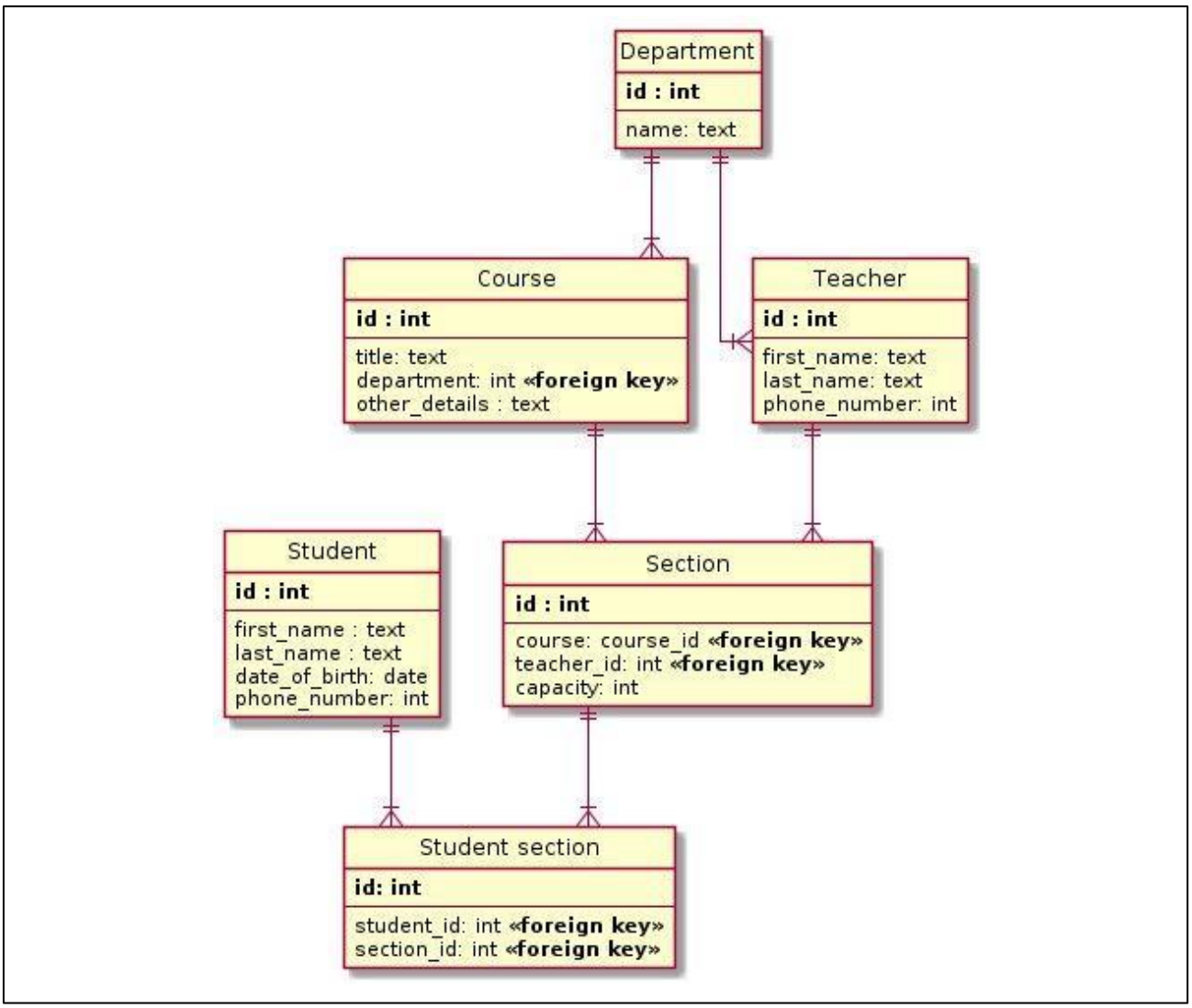

Рисунок 6.5 – Схема реляційної БД.

Система керування вмістом (англ. ContentManagementSystem, CMS) – це система керування контентом, набір скриптів для створення, редагування та управління контентом сайту. На професійному жаргоні CMS також називають «двигун». Прикладами CMS є WordPress, Joomla,

Існують сотні, а може, навіть й тисячі доступних CMS-систем. Завдяки їх функціональності ці системи можна використовувати в різних компаніях. Незважаючи на широкий вибір інструментальних і технічних засобів, наявних в CMS, існують загальні для більшості типів систем характеристики.

Системи управління веб-сайтом часто розраховані на роботу у певному програмному середовищі. Наприклад, система MediaWiki, під управлінням якої працює Вікіпедія, написана мовою програмування PHP і зберігає вміст і налаштування у базі даних типу MySQL або PostgreSQL; тому для її роботи потрібно, щоб на сервері, де вона розміщена, були встановлені: веб-сервер (Apache, IIS чи інший), підтримка PHP та системи керування базами даних MySQL або PostgreSQL, а також, в разі необхідності, додаткові програми для обробки зображень чи математичних формул. Такі вимоги є досить типовими для відкритих СКВ.

Якщо раніше більшість сайтів були статичними і вимагали внесення правок в їх вміст вручну, зараз динаміка розвитку проектів вимагає готовності швидко реагувати на зміни і впроваджувати їх з максимальною оперативністю. При цьому не всі користувачі хочуть або можуть собі дозволити звертатися до розробників, особливо якщо сайт вимагає постійної роботи над ним.

У свою чергу, системи керування контентом дозволяють користувачам, які не мають навичок розробки сайтів і знання мов програмування, самостійно працювати над створенням і зміною сайту.

Як правило, CMS використовують для таких сайтів:

- блог, форум (WordPress, phpBB, vBulletin);
- інтернет магазин (Magento, OpenCart, osCommerce);
- соціальні мережі (InstantCMS, Social Engine);
- персональні сайти (WordPress, Monstra);
- корпоративні сайти (Joomla, Drupal);
- портали (DLE, Drupal).

Фреймворк (Framework, каркас, платформа, структура, інфраструктура) – інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має низку обмежень, що задають правила створення структури проекту та написання коду.

Програмний фреймворк (softwareframework) – це готовий до використання комплекс програмних рішень, який охоплює дизайн, логіку та базову функціональність системи або підсистеми. Відповідно, програмний фреймворк

може містити в собі також допоміжні програми, деякі бібліотеки коду, скрипти та загалом все, що полегшує створення та поєднання різних компонентів великого програмного забезпечення чи швидке створення готового і не обов'язково об'ємного програмного продукту. Побудова кінцевого продукту відбувається, зазвичай, на базі єдиного API.

Одна з головних переваг при використанні *каркасних застосунків* полягає в тому, що такі програми мають стандартну структуру. Каркаси застосунків стали популярними з появою елементів інтерфейсу, які мали тенденцію до реалізації стандартної структури для додатків. З їх використанням стало набагато простіше створювати засоби для автоматичного створення графічних інтерфейсів, оскільки структура внутрішньої реалізації коду програми стала відома заздалегідь. Для забезпечення каркаса, зазвичай, використовують підходи об'єктно- орієнтованого програмування, наприклад, частини програми можуть успадковуватися від базових класів фреймворку.

Якщо у програміста стоїть завдання створити сайт, йому необхідно відразу ж визначити подальшу стратегію роботи. Є три шляхи розробки, кожен програміст може вибрати той, який найбільше підходить під його вміння.

Можна написати необхідний *вихідний код з нуля*. Головною перевагою цього варіанта є його варіативність – практично ніяких обмежень, можна реалізувати будь-який задуманий функціонал, потрібні лише певні вміння. Головним мінусом можна назвати трудомісткість процесу, тимчасові витрати. Також доведеться докласти дуже багато зусиль для ретельного тестування отриманого продукту, а саме: доведеться знайти все його вади, щоб створити ідеальний веб-проект.

Використання фреймворків. Існують певні обмеження, якщо проводити паралелі з попереднім способом. Існує основа, в яку потрібно додати певну кількість необхідних компонентів. Даний варіант є рентабельним лише для тих, хто хоч трохи розбирається в програмуванні, бо без певної кількості знань виконати поставлене завдання правильно практично неможливо. Для людей, які не можуть скористатися наведеними способами, є альтернативний варіант.

Використання *готової CMS*. Даний варіант є ідеальним для людей, які мало розуміють в сегменті веб-розробки. Ви зможете оперативно створити сайт, відповідний вашим вимогам. Є можливість вносити необхідні корективи через адміністративну панель. Але даний підхід не користується особливою популярністю – головним мінусом є величезна кількість обмежень.

Система керування версіями (СКВ, англ. sourcecodemanagement, SCM) – програмний інструмент для керування версіями одиниці інформації: початкового коду програми, скрипту, веб-сторінки, вебсайту, 3D-моделі, текстового документа тощо.

Система контролю версій – це система, що записує зміни у файл або набір файлів протягом деякого часу, так що ви зможете повернутися до певної версії

пізніше. Існують два основні типи систем керування версіями: з централізованим сховищем та розподіленим.

Більшість користувачів як один з методів контролю версій застосовують копіювання файлів в окрему директорію (можливо навіть директорію з відміткою за часом, якщо вони достатньо розумні). Даний підхід є дуже поширеним завдяки його простоті, проте він, неймовірним чином, схильний до появи помилок. Можна легко забути в якій директорії ви знаходитеся і випадково змінити не той файл або скопіювати не ті файли, які ви хотіли.

Щоб справитися з цією проблемою, програмісти давно розробили локальні СКВ, що мають просту базу даних, яка зберігає всі зміни в файлах під контролем версій (рис. 6.6).

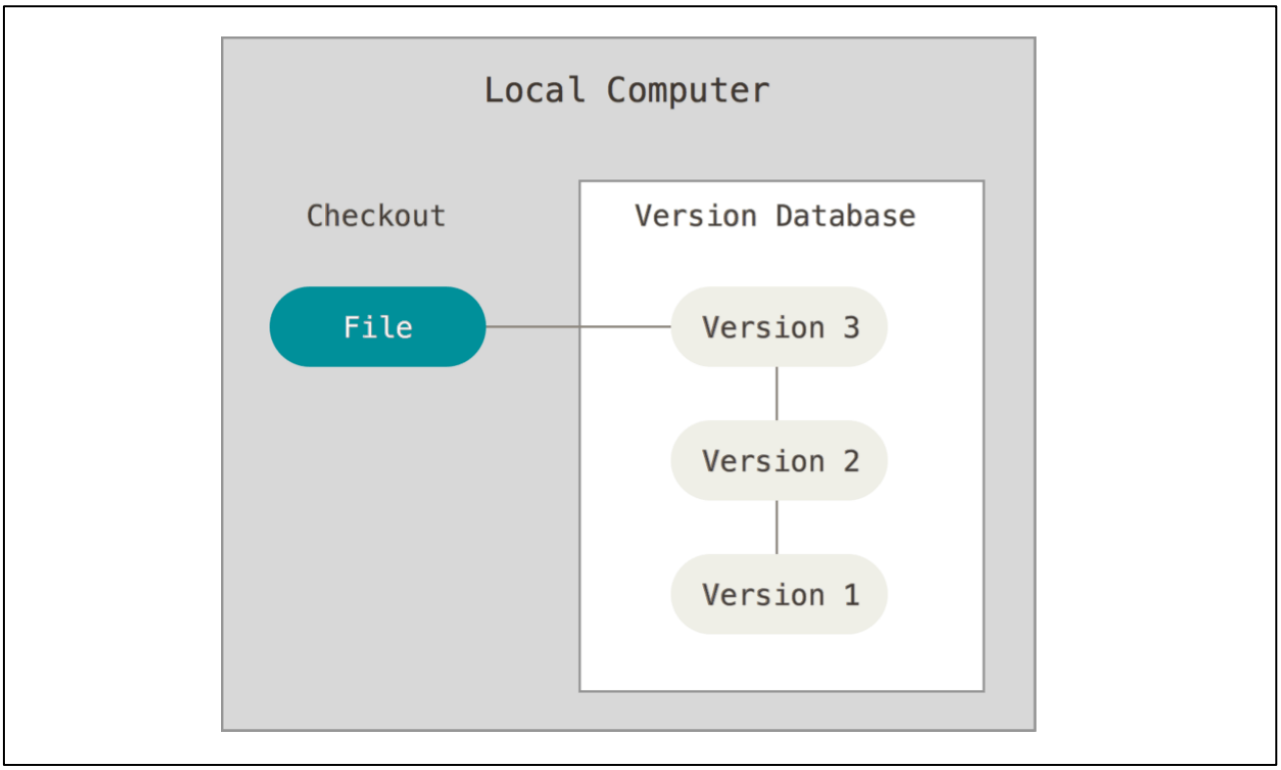

Рисунок 6.6 – Схема локальної системи контролю версій.

Одним з найбільш поширених інструментів СКВ була система під назвою RCS, яка досі поширюється з багатьма комп'ютерами сьогодні. RCS зберігає набори «латок» (тобто, відмінностей між файлами) в спеціальному форматі на диску; він може заново відтворити будь-який файл, як він виглядав, в будь-який момент часу, шляхом додавання всіх «латок».

Наступним важливим питанням, з яким стикаються люди, є необхідність співпрацювати з іншими розробниками. Щоб справитися з цією проблемою, були розроблені централізовані системи контролю версій (ЦСКВ) (рис. 6.7). Такі системи, як CVS, Subversion і Perforce, мають єдиний сервер, який містить всі версії файлів, та деяке число клієнтів, які отримують файли з центрального сервера.

Протягом багатьох років, це було стандартом для систем контролю версій. Такий підхід має безліч переваг, особливо над локальними СКВ. Наприклад, кожному учаснику проєкту відомо, певною мірою, чим займаються інші. Адміністратори мають повний контроль над тим, хто і що може робити. Набагато легше адмініструвати ЦСКВ, ніж мати справу з локальними базами даних для кожного клієнта.

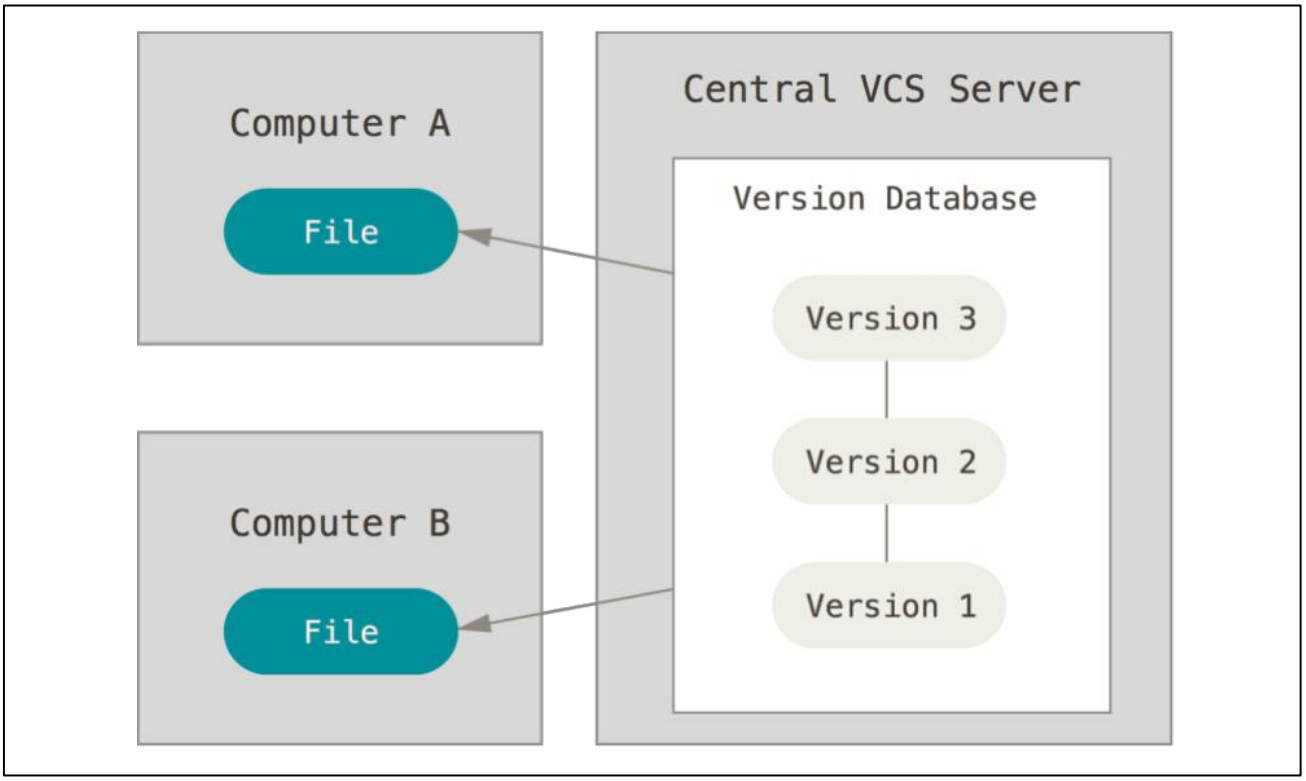

Рисунок 6.7 – Схема централізованої системи контролю версій.

В децентралізованих системах контролю версій (рис. 6.8) клієнти не просто отримують останній знімок файлів репозиторія: натомість вони є повною копією сховища разом з усією його історією. Таким чином, якщо «вмирає» який-небудь сервер, через який співпрацюють розробники, будь-який з клієнтських репозиторіїв може бути скопійований назад до сервера, щоб відновити його. Кожна копія дійсно є повною резервною копією всіх даних.

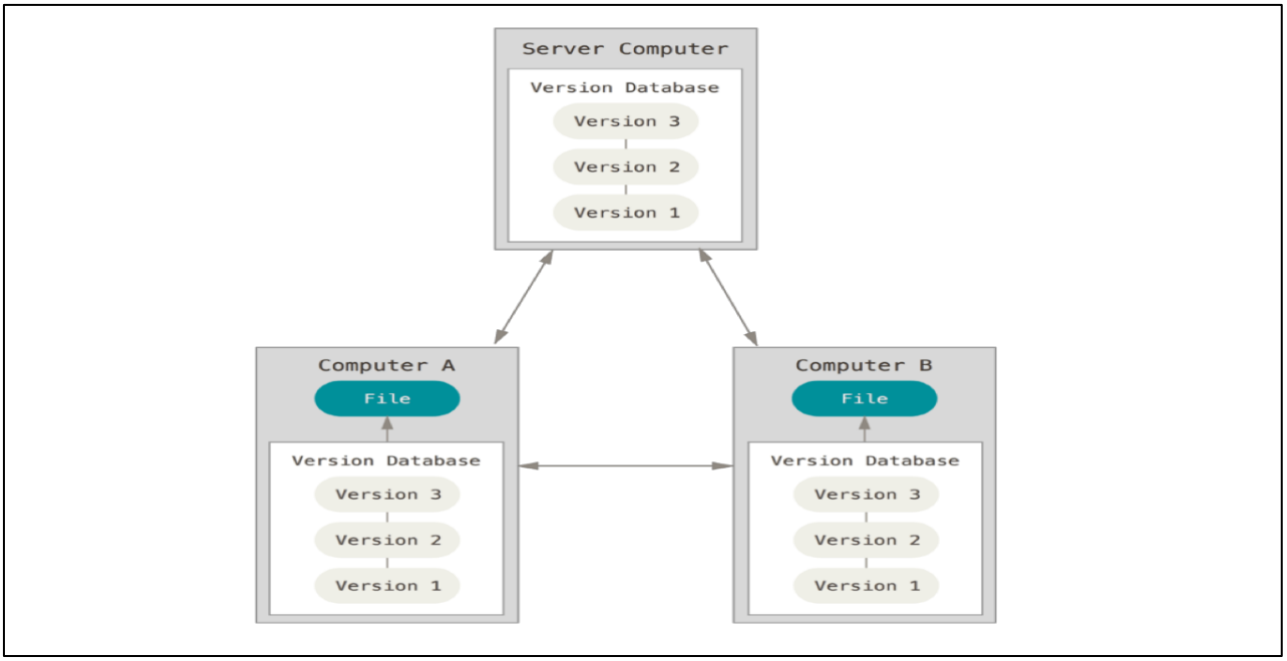

Рисунок 6.8 – Схема децентралізованої системи контролю версій.

Більш того, системи добре взаємодіють з декількома віддаленими репозиторіями, так що ви можете співпрацювати з різними групами людей, застосовуючи різні підходи в межах одного проекту одночасно. Це дозволяє налаштувати декілька таких типів робочих процесів, як ієрархічні моделі, які неможливі в централізованих системах.

Розподілена система контролю версії (Distributed Version Control System, DVCS) – система, яка використовує замість моделі клієнт – сервер, розподілену модель зберігання файлів. Така система не потребує сервера, адже всі файли знаходяться на кожному з комп'ютерів. До розподілених систем відносять: Git, Mercurial, Bazaar, Monotone, Codeville, BitKeeper.

6.2 Розробка мобільних застосунків

У сучасному світі мобільні пристрої стали невід'ємною частиною нашого життя. Вони допомагають нам спілкуватися, працювати, навчатися, розважатися та вирішувати безліч інших завдань. Але можливо це завдяки додаткам, що встановлюються на ці пристрої.

Розуміння принципів розробки мобільних застосунків допомагає не лише розуміти, як працюють улюблені програми, але й відкриває можливості для креативного використання технологій від освіти до бізнесу.

Середньостатистичний користувач смартфона проводить близько 3-5 годин на день використовуючи його більше 200 разів на день, користуючись різними мобільними додатками. Станом на 2024 рік, кількість доступних застосунків у основних магазинах додатків перевищує 5 мільйонів.

Мобільний застосунок (додатки) – це програма, розроблена спеціально для мобільних пристроїв, таких як смартфони, планшети або смарт-годинники, що

дозволяють користувачам виконувати різноманітні завдання, задовольняючи свої потреби (від комунікації та розваг до управління фінансами та освіти).

Спочатку мобільні застосунки використовувалися для швидкої перевірки електронної пошти, але їх високий попит призвів до розширення їх призначень і в інших областях, таких як ігри для мобільних телефонів, GPS, спілкування, перегляд відео та користування Інтернетом. В мобільних програмах задіяні функціональні можливості:

- робота з камерою;
- встановлення з'єднання в мережах стільникового зв'язку;
- можливість підключення сумісних пристроїв через Bluetooth;
- може використовуватися Wi-Fi для входу в інтернет;
- задіяна GPS-навігація.

Існує кілька типів класифікації мобільних додатків, що залежать від критерію оцінки.

За цільовою аудиторією:

- *Корпоративні*: призначені для внутрішнього використання у компаніях.
- *Споживчі*: розроблені для кінцевих користувачів, спрямовані на розваги, соціальні мережі тощо.

За операційною системою (платформою):

- *iOS*: для пристроїв Apple.
- *Android*: для смартфонів і планшетів на базі операційної системи Android.
- Гібридні: розроблені для роботи на різних платформах.

За функціональністю:

- *Ігрові*: орієнтовані на розваги та геймінг.
- *Продуктивність*: допомагають у плануванні, роботі та організації.
- *Спілкування*: включають месенджери, соціальні мережі та інструменти для спілкування.

За технологією:

- *Нативні*: розроблені спеціально для однієї конкретної платформи, використовуючи мови програмування, такі як Swift або Java. Мають високу продуктивність і доступ до всіх функцій пристрою;
- *Веб-додатки*: запускаються у веб-браузері на мобільному пристрої і не потребують встановлення через магазин додатків, але можуть мати обмежений доступ до функцій пристрою.
- *Гібридні*: комбінують елементи нативних та веб-додатків. Написані на мовах, таких як HTML, CSS і JavaScript, і збираються в аплікації за допомогою фреймворків, таких як Ionic або React Native.

Сучасний мобільний додаток повинен відповідати наступним вимогам:

1. *Інтуїтивний інтерфейс*: зручний та легкий у використанні інтерфейс дозволяє користувачам швидко орієнтуватися у додатку без зайвих зусиль.
2. *Безпека*: захист особистої інформації користувачів є критично важливим. Мобільні додатки повинні забезпечувати високий рівень безпеки та конфіденційності.
3. *Адаптивність*: додатки повинні бути адаптованими до різних типів пристроїв, розмірів екранів та орієнтацій.
4. *Швидкодія*: висока продуктивність та швидке завантаження допомагають забезпечити позитивний досвід користувача.
5. *Оновлення*: регулярні оновлення для виправлення помилок, вдосконалення функціоналу та забезпечення сумісності з новими версіями операційних систем.

Розробка мобільного застосунку це створення цифрової системи, здатної ефективно функціонувати в системі мобільних пристроїв. Його основа – це архітектура. Вона визначає, як працюють і пов'язані між собою різні частини програми. Користувач не бачить архітектуру, але вона дуже важлива.

Ефективна архітектура робить додаток надійним, швидким і готовим до оновлень. Це як проект будівлі або міцний фундамент для будинку. Вибір правильної архітектури впливає на швидкість роботи застосунку, його здатність адаптуватися до нових функцій або збільшення даних. Популярні архітектурні підходи:

Монолітна архітектура: у цьому підході всі компоненти застосунку інтегровані в один єдиний блок. Це дозволяє спрощувати розгортання, але може викликати проблеми з масштабованістю та оновленням. Наприклад, бухгалтерський застосунок для бізнесу, включає функції для ведення фінансових записів, створення звітів, управління рахунками, обліку витрат і прибутків. Монолітна архітектура в цьому випадку означає, що весь функціонал програми знаходиться в одному кодовому блоці, який розгортається як єдине ціле (див. рис. 6.9). Це означає, що всі функції інтегровані в одну програму.

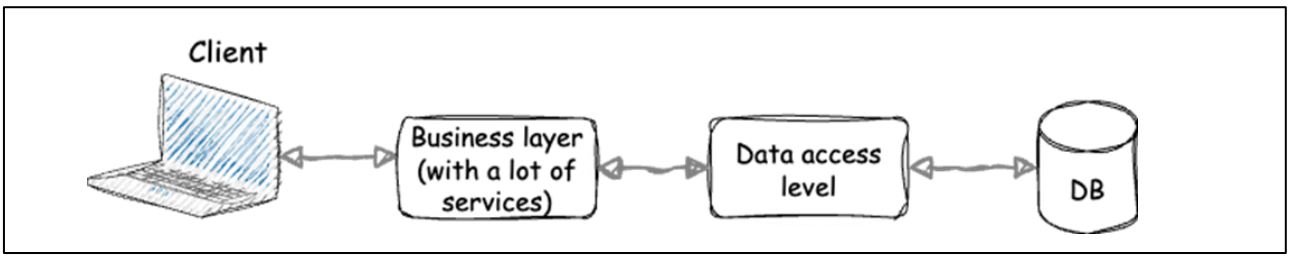

Рисунок 6.9 – Схема монолітної архітектури додатку.

Мікросервісна архітектура: застосунок розбивається на незалежні сервіси, кожен з яких відповідає за окрему функцію. Це підвищує гнучкість і дозволяє легко масштабувати окремі компоненти, може ускладнити управління. Наприклад, інтернет-магазин в такій архітектурі може мати окремий мікросервіс

для управління товарами, інший для обробки замовлень і ще один для роботи з користувачами. Всі вони працюють незалежно один від одного, але разом забезпечують функціонування магазину (див. рис. 6.10).

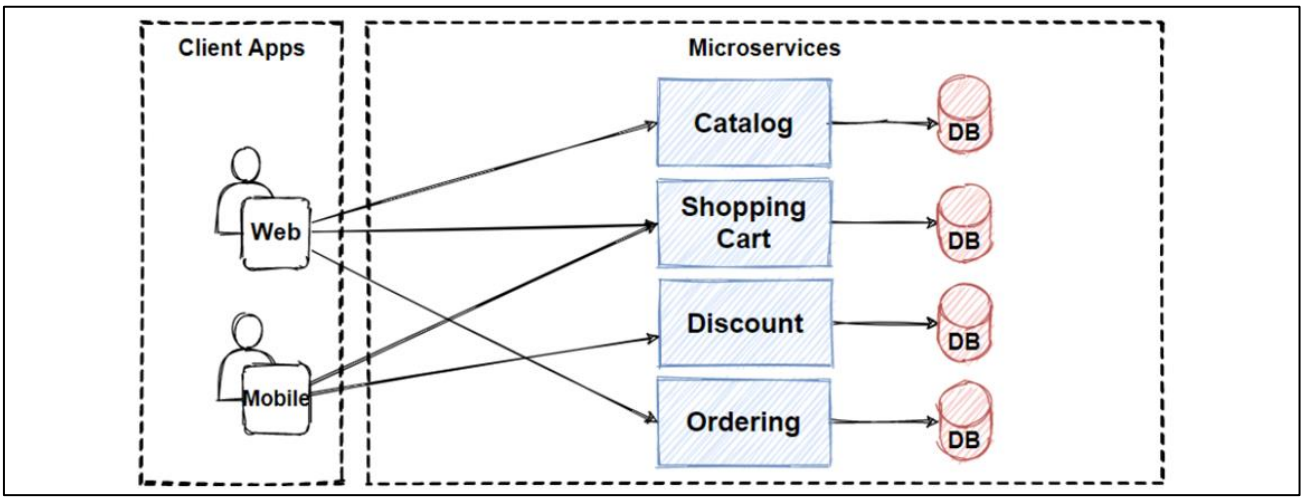

Рисунок 6.10 – Схема мікросервісної архітектури додатку.

Сервіси на базі контейнерів: використання контейнерів дозволяє ізолювати різні компоненти застосунку та їх залежності, що спрощує розгортання та масштабування. Наприклад, велика корпорація може мати різні системи для управління фінансами, клієнтами, товарами, і всі ці системи можуть взаємодіяти через стандартизовані сервіси (див рис. 6.11).

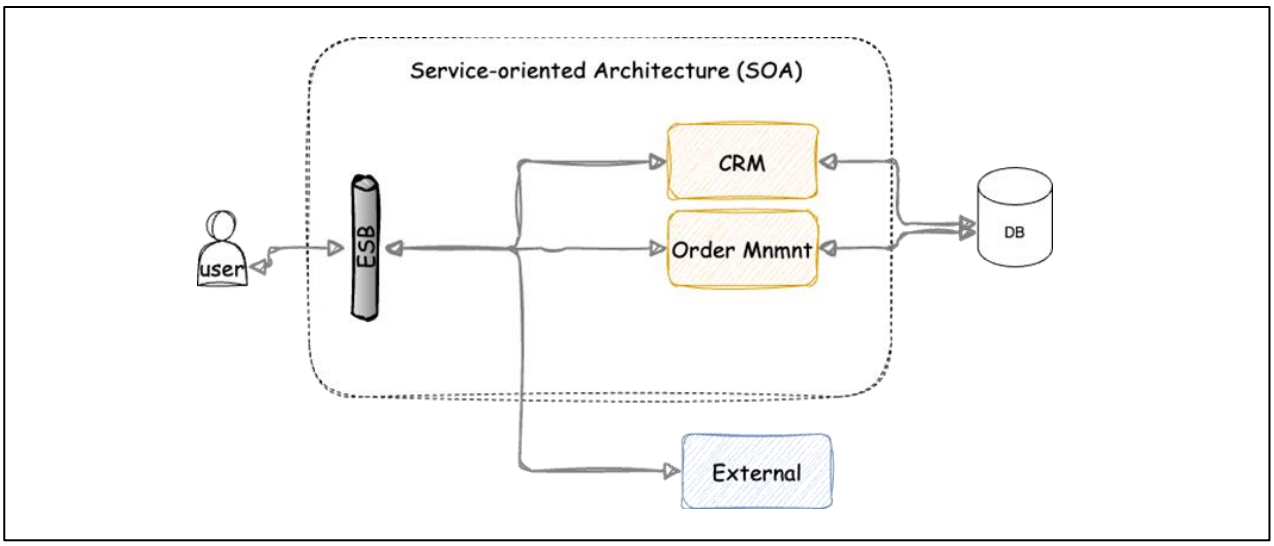

Рисунок 6.11 – Схема архітектури додатку на основі контейнерів із сервісами.

Клієнт-серверна архітектура: складається з двох частин: клієнт (зазвичай це те, що бачить користувач – наприклад, браузер) і сервер (місце, де зберігаються дані і виконується основна логіка). Припустимо банківська програма складається з мобільного додатку на телефоні (клієнт) а сервер – це комп'ютери банку, де зберігаються рахунки та обробляються транзакції (див. рисю 6.12).

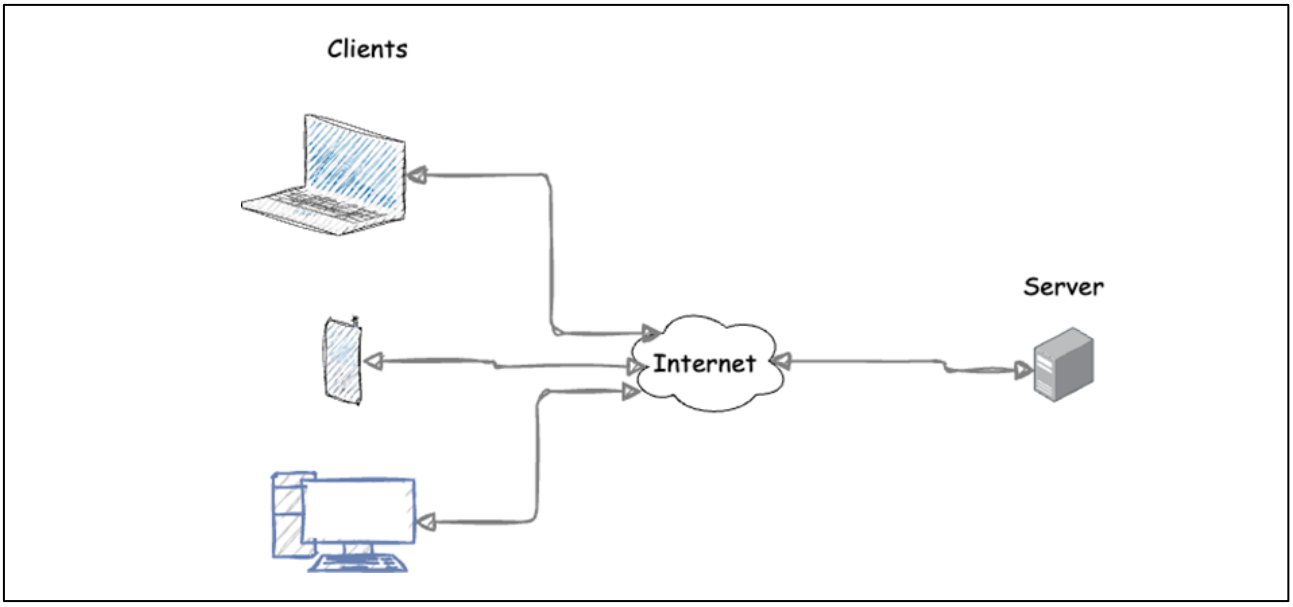

Рисунок 6.12 – Схема клієнт-серверної архітектури додатку.

Вибір архітектури залежить від багатьох факторів: масштабу проєкту, кількості користувачів, необхідності масштабування та розширення, і навіть технологічних можливостей команди. Кожен тип архітектури має свої плюси і мінуси, і найкраща архітектура – це та, що відповідає конкретним вимогам вашої системи. Архітектурні шаблони – це готові рішення для повторюваних проблем, які зустрічаються при розробці програмного забезпечення і шаблони допомагають організовувати структуру і взаємодію компонентів в кожному типі архітектури

Прогресивний веб-застосунок (англ. Progressive Web App, PWA) – веб-застосунок, який є гібридом звичайної веб-сторінки (чи веб-сайту) та мобільного застосунку. Завдяки підходу PWA можна уникнути необхідності в окремій розробці сайту та додатку для смартфону. Технологія трансформує сайт у застосунок, стираючи головні відмінності. Завдяки їй користувач може додати ярлик на головний екран, увімкнути push-повідомлення, працювати офлайн. Тобто, PWA дає користувачу змогу користуватись сайтом як додатком, не завантажуючи його на девайс.

Зовні продукт виглядає як додаток, проте займає на смартфоні мало місця, оскільки майже усю інформацію зберігає у хмарі. Як і звичайний сайт, він автоматично оновлюється, не вимагаючи будь-яких дій з боку споживача. Веб-застосунок, встановлений на смартфоні, працює швидше, ніж звичайний сайт, надає доступ офлайн, отримує частковий доступ до апаратної частини тощо.

Мобільні застосунки є надзвичайно важливою частиною сучасних технологій та бізнесу. Лекція показала, що мобільні технології продовжують розвиватися та впливати на різні аспекти нашого життя. Від розробки застосунків для бізнесу і розваг до інноваційних рішень у сфері здоров’я, освіти та розумного дому. Мобільні додатки відкривають безмежні можливості завдяки інтеграції різних датчиків, сучасним мовам програмування та інноваційним технологіям. Ці

перспективи створюють сприятливі умови для постійного впровадження нових технологій, що допомагає вдосконалювати існуючі застосунки.

6.3 Розробка з низьким кодом або без коду

Платформи з низьким кодом або без коду (англ. Low-Code/No-Code platform, абрв. – LCNC platform) здійснили революцію в розробці програмного забезпечення. LCNC платформи дозволяють суттєво знизити рівень технічної підготовки розробників програмного забезпечення (ПЗ) та створювати продукти значно швидше.

Алгоритми та принципи створення інформаційних систем та програмних продуктів суттєво еволюціонували з моменту їх появи. Можна виокремити п’ять поколінь:

- 1943 р. – *перше покоління*, розробка ПЗ шляхом написання коду у двійковій формі (Electronic Numerical Integrator and Computer);
- 1949 р. – *друге покоління*, розробка ПЗ шляхом написання коду за допомогою асемблерних мов, які спростили машинний код;
- 1957 р. – *третє покоління*, розробка ПЗ шляхом написання коду за допомогою процедурних мов високого рівня (Fortran, Algol, COBOL);
- 1965 р. – *четверте покоління*, розробка ПЗ шляхом написання коду за допомогою об'єктно-орієнтованого програмування з появою мов Simula та Smalltalk;
- 2014 р. – *п’яте покоління*, вихід безкоштовної версії Outsystemsплатформи для розробників (LCNC платформи), що побачила світ ще у 2001 р., заснована з ідеєю пришвидшення цифрової трансформації.

Візуальні інструменти розробки Microsoft Access та Visual Basic у 80-90х роках XX століття, започаткували фундаментальні зміни в підході до створення програм, відходячи від громіздкого коментування коду до графічного середовища. За допомогою цих інструментів розробники вже тоді створювали прості застосунки майже без написання коду, і це заклало основу для появи більш досконалих платформ.

Розвиток хмарних обчислень та поява моделі бізнесу «програмне забезпечення як послуга» (англ. Software As A Service, SaaS) у 2000-х роках комулятивно збільшив розвиток LCNC платформ. Згодом з’явилися інструменти з drag-and-drop (перетягування елементів) функціональністю та попередньо створеними компонентами, що значно спростило створення застосунків для нетехнічних користувачів.

Після 2010 року LCNC платформи активно розвивалися завдяки сукупності технологічних досягнень, включно зі штучним інтелектом, API та орієнтацією на мобільні пристрої. Нові постачальники, такі як Mendix, OutSystems та Bubble, запропонували потужні й масштабовані рішення для створення застосунків будь-якої складності.

Поява LCNC платформ започаткувала зміну парадигми у сфері розробки програмного забезпечення. Компанія «Forrester» у 2014 році запровадила термін *Low-code development platform* (LCDP), а у 2016 році компанія «Gartner» ввела у вжиток термін *Low-code application platform* (LCAP). У 2017 році генеральний директор GitHub Кріс Ванстрат (англ. ChrisWanstrath) заявив, що майбутнє програмування – це відсутність програмування взагалі.

Відтоді почала формуватись тенденція до мінімізації обсягу коду та зменшення початкових інвестицій у налаштування, навчання та розгортання програмного забезпечення. Бізнес прагнув до скорочення часу виходу на ринок, підвищення гнучкості й зменшення залежності від IT-команд.

LCNC платформи пропонують інтуїтивні інтерфейси з підтримкою перетягування елементів (drag-and-drop), що значно розширюють перелік їх застосовування: моделі AI (Artificial Intelligence), автоматизовані робочі процеси, безшовну інтеграцію між різними платформами, корпоративні системи, веб-застосунки, програми під усі операційні системи, тощо.

В основі LCNC платформ лежить принцип чорного ящика (англ. black box). Ця концепція описує систему з певним рівнем абстракції – нехтуючи тим як працює «чорний ящик», натомість концертуючись на вхідних та вихідних сигналах. Оскільки розробка ПЗ має багато часто-повторюваних етапів, LCNC платформа приймає на вхід (вхідний сигнал) значення параметрів об’єкта системи, що визначають вимоги до його стану, і як результат сам об’єкт (чорний ящик) створює вихідний сигнал максимально швидко. При цьому не варто перестворювати сам об’єкт.

Отже, платформи з низьким кодом або без коду є перспективним напрямком розвитку галузі інформаційних технологій. Вони пропонують інтуїтивні інтерфейси, які дозволяють людям із мінімальними знаннями програмування створювати, збирати та розгортати програмні рішення, а головне, значно зменшити витрати на розробку.

LCNC платформи є сильною альтернативою класичній розробці (написанні коду), в повній або частковій мірі. У майбутньому інтеграція зі штучним інтелектом і машинним навчанням продовжить визначати вектор розвитку, забезпечуючи створення більш автоматизованих, персоналізованих і розумних застосунків, які адаптуються до індивідуальних потреб і вподобань користувачів.

7. ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ СИСТЕМИ

7.1 Передумови виникнення

Інтелектуальні інформаційні системи – це інформаційні системи, здатні моделювати процеси прийняття рішень, навчатися на основі даних, адаптуватися до змін середовища та надавати рекомендації або здійснювати дії з високим рівнем автономності. Їхнє формування стало результатом злиття кількох технологічних і наукових напрямів. Зокрема, розвиток обчислювальної техніки (збільшення потужності мікропроцесорів, поява персональних комп'ютерів та серверів), алгоритмічний прорив (створення методів машинного навчання, логічного виведення, статистичного аналізу) та розвиток бізнесу (потреба швидко аналізувати дані для підвищення ефективності управління).

Перший етап розвитку почався в другій половині XX-го століття (1950–1980 рр.), і визначав перші кроки до формування експертних систем:

- 1950 рік – Алан Тюрінг публікує статтю «Computing Machinery and Intelligence» і пропонує тест Тюрінга для оцінки інтелекту машин;
- 1956 рік – конференція в Дартмутському коледжі (США), де Джон Маккарті, Марвін Мінський, Клод Шеннон і Натаніель Рочестер вводять термін *Artificial Intelligence*;
- 1965 рік –Джозеф Вейценбаум створює програму ELIZA, яка імітує діалог із психотерапевтом, використовуючи патерн-матчинг;
- 1972 рік – Едвард Шортліфф розробляє MYCIN – експертну систему для діагностики бактеріальних інфекцій та вибору антибіотиків. Вона містила близько 450 правил і показала кращі результати, ніж деякі лікарі;
- 1979 рік – компанія Digital Equipment Corporation запускає XCON (eXpert CONfigurer) для автоматичної конфігурації VAX-серверів.

В цей період системи були вузькоспеціалізованими, базувалися на знаннях експертів, реалізованих у вигляді логічних правил, і не використовували машинне навчання у сучасному розумінні.

Другий етап тривав втричі менше (1980–1990 рр.) і був спрямований на інтеграцію з базами даних і DSS:

- 1980-ті – зростання популярності Decision Support Systems (DSS), які поєднують бази даних і математичне моделювання для допомоги керівникам;
- 1983 рік – вихід Lotus 1-2-3 – це електронна таблиця, що стала стандартом бізнес-аналітики;
- 1986 рік – компанія IBM запускає Expert System Environment (ESE) для промислових застосувань експертних систем;
- 1989 – Ральф Кімбалл і Вільям Інмон формулюють концепцію Data Warehouse, яка передбачала централізоване сховище для аналітики.

Цей період характеризується переходом від ізольованих експертних систем до інтегрованих рішень з можливістю роботи з великими масивами даних.

Третій етап (1990–2000 рр.) характерний для появи Business Intelligence:

- 1993 рік – аналітик Howard Dresner популяризує термін Business Intelligence;
- З'являються комерційні BI-платформи: Cognos, BusinessObjects, MicroStrategy;
- Розвиваються OLAP-куби для швидкого багатовимірного аналізу даних;
- Інтелектуальні системи починають включати інструменти прогнозування на основі статистики.

Четвертий етап (після 2010 року) це період великих обсягів даних (Big Data) та AI в тому розумінні, що ми його знаємо:

- 2010-ті роки – масовий розвиток Big Data-технологій (Hadoop, Spark) та хмарних обчислень (AWS, Azure, Google Cloud);
- 2012 рік – прорив у глибинному навчанні завдяки мережі AlexNet (Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever), яка перемогла на конкурсі ImageNet;
- 2020-ті роки – поява мовних моделей великого масштабу (GPT-3, GPT-4, Claude, Gemini), інтеграція AI в робочі процеси, створення автоматизованих агентів.

7.2 Business Intelligence

Сучасні потреби бізнесу та науки впливають на розвиток технологій для аналізу даних. Business Intelligence (BI) охоплює методології, процеси та технології, що перетворюють необроблені дані на цінну інформацію.

Business Intelligence (BI) – це комплекс технологій, процесів і методологій для збору, обробки та аналізу даних з метою підтримки прийняття управлінських рішень. Він дозволяє:

- Отримувати звіти в реальному часі.
- Виявляти тренди та аномалії.
- Оптимізувати бізнес-процеси.

Business Intelligence (BI) термін, що узагальнює концепцію та методи, процеси та технології, що систематизують дані, перетворюючи їх на цінну інформацію, яка допомагає в швидкому й ефективному прийнятті рішень (стратегічних, тактичних і операційних). BI інтегрує технологічні, комерційні та правові аспекти, сприяючи досягненню бізнес-цілей. Основою BI є системи підтримки рішень, які аналізують дані з внутрішніх і зовнішніх джерел, перетворюючи їх у корисні аналітичні висновки.

BI-системи надають інструменти для створення звітів і дашбордів (dashboards) із ключовими показниками ефективності – метриками, що допомагають оцінювати динаміку процесів і оптимізувати діяльність. Вони застосовуються у різних галузях для аналізу поведінки споживачів, сегментації ринку, управління ціноутворенням та вдосконалення виробничих ланцюгів. BI-системи – це «вітрини даних».

Було виявлено стійку тенденцію до зростання у дослідженнях методів і технік Business Intelligence з помітним прискоренням у після 2020 року. В період 2010-2017 рр. в середньому, на рік припадало, близько 100-150 публікацій, коли зараз більш 500 публікацій на рік, і цей показник невпинно зростає.

Попри значну кількість публікацій, деякі напрями BI залишаються малодослідженими, зокрема: інтеграція BI з конкурентною аналітикою і його роль у сфері управління персоналом, зв’язок BI з технологіями глибинного аналізу (data mining) та вплив на прийняття управлінських рішень.

Конкурентна аналітика – це систематичний підхід до збору та аналізу інформації про ринок, конкурентів і внутрішні бізнес-процеси, що допомагає організаціям адаптуватися до змін і ухвалювати стратегічні рішення. Конкурентна аналітика допомагає компаніям адаптуватися до змін ринку, аналізувати стратегії конкурентів і вибудовувати ефективні бізнес-моделі. Дослідження цих аспектів може розширити можливості BI у розвитку організацій.

Також, подальший розвиток BI пов’язаний із посиленням інтеграції зі штучним інтелектом, автоматизацією бізнес-процесів та розширенням можливостей самостійного аналізу даних користувачами. Ще одним перспективним напрямом є підвищення відкритості та доступності BI-інструментів, що сприятиме їх активнішому впровадженню в різних сферах діяльності.

7.3 Науково-дослідницький аналіз

Сфера науково-дослідницького аналізу даних (англ. Data Science) дуже популярна в наш час. Саме її стрімкий розвиток передував появі в загальному доступі інструментів штучного інтелекту (англ. Artificial intelligence, AI).

Data Science – це галузь, що займається збором, обробкою, аналізом, та інтерпретацією даних, з метою здобуття нових знань та розуміння явищ в певній сфері. Вона поєднує методології, алгоритми та інструменти з різних дисциплін, включаючи статистику, математику, інформатику та інженерію.

Data Science є ключовим елементом в сучасному світі, де дані грають важливу роль у прийнятті рішень, плануванні, та стратегічному розвитку різних галузей. Із розвитком технологій та зростанням об'ємів даних, значення Data Science тільки збільшується, тому вона стає все більш популярною серед фахівців різних напрямків.

Життєвий цикл проєкту Data Science – це важлива тема для розуміння того, як правильно підходити до роботи з даними, який складається з п’яти послідовних кроків (див. рис. 7.1). Він складається зі збору даних, очищення та підготовка даних, дослідження даних, створення моделі (гіпотези) та інтерпретації (застосування).

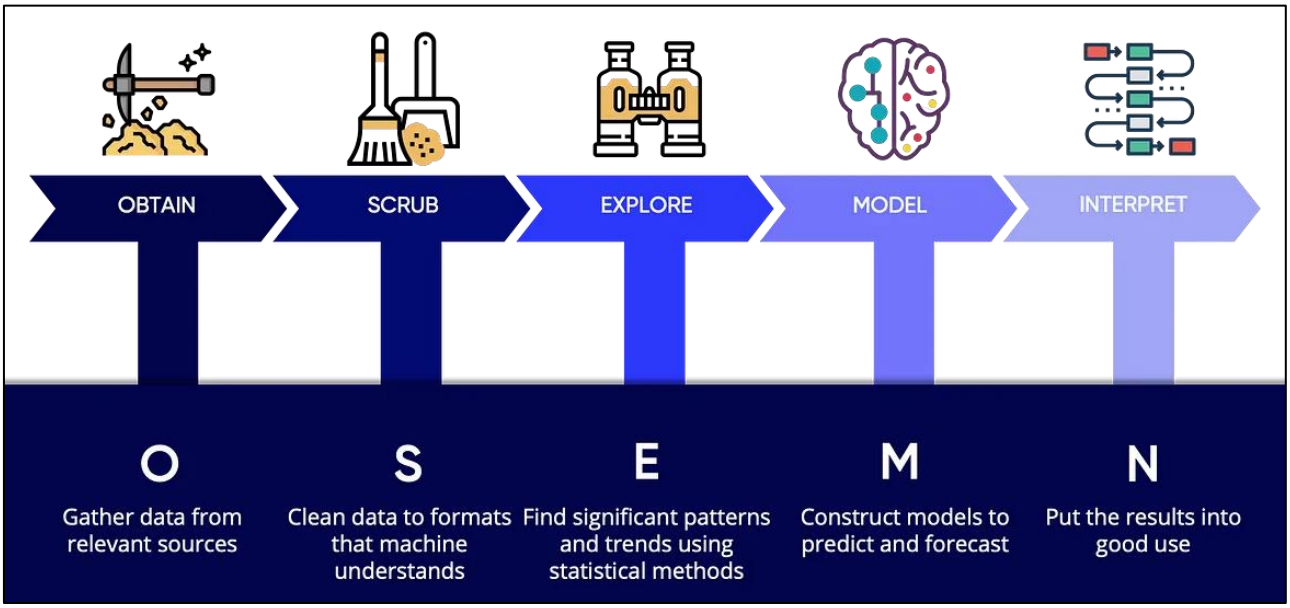

Рисунок 7.1 – Життєвий цикл Data Science проєкту.

Збір даних – перший етап життєвого циклу проекту Data Science, на якому важливо отримати дані певного формату та структури. Цей етап є важливим, оскільки точність і достовірність будь-якого аналізу даних залежить від якості вхідних даних. Зазвичай дані збираються з різних джерел, таких як бази даних, датчики, API тощо. Залежно від проєкту, дані можуть бути у форматі структурованого або неструктурованого тексту, зображень, відео або звуку.

Наприклад, якщо ми хочемо побудувати модель передбачення ціни нерухомості на підставі її характеристик, то ми можемо зібрати дані з баз даних нерухомості, від агентів нерухомості, а також з соціальних мереж, де користувачі діляться інформацією про купівлю та продаж нерухомості. Дані можуть бути в різних форматах, таких як CSV, JSON або XML, і їх можна зібрати за допомогою різних інструментів та бібліотек, наприклад, Python (мова програмування) бібліотеки *pandas* або *BeautifulSoup* для збору даних з веб-сторінок.

Очищення та підготовка даних – один з найважливіших етапів у життєвому циклі проекту Data Science, на якому зібрані данні очищуються від дублікатів, помилок та приводиться до однорідної структури, створюючи датасет (англ dataset) – колекція (набір) однотипних даних. Незалежно від того, наскільки у нас хороша модель, більшість залежить від нашого датасету, і є хороший вислів: *сміття на вхід - сміття на вихід*. Очищення даних означає видалення або коригування неточностей, помилок, дублікатів, відсутності значень та інших некоректних даних. На цьому етапі можна проводити різні операції, наприклад:

- *Видалення дублікатів*: у великих наборах даних можуть бути дублікати записів, що може призвести до перекручення результатів. Наприклад у таблиці можуть бути стовпці *User* та *Name*, проте у кожному з них буде зберігатись одна й та ж інформація - ім'я користувача. Видалення дублікатів забезпечує точність даних і зменшує їх об'єм.
- *Видалення відсутніх значень*: відсутні значення в даних можуть виникати з різних причин, включаючи помилки вводу, відмову учасників дослідження відповідати на певні питання або недоступність даних. Відсутні значення можуть бути замінені середніми значеннями або іншими вирахованими значеннями.
- *Кодування категоріальних даних*: багато даних містять категоріальні дані, які потрібно закодувати в числові значення для подальшої обробки. Це можна зробити за допомогою різних методів, таких як метод one-hot encoding або label encoding.
- *Нормалізація даних*: дані можуть містити значення, які відрізняються за порядком величин, наприклад, вік та дохід. Для того, щоб забезпечити рівні ваги для всіх факторів, дані можуть бути нормалізовані за допомогою різних методів.

Даний перелік не є вичерпним, проте описує основні проблеми, з якими ви можете зіткнутись.

Дослідження даних (англ. Exploratory Data Analysis, EDA) – це перший крок у будь-якому проєкті зі статистичним моделюванням та машинним навчанням. Його метою є дослідження даних, що входять до дослідження, їхнього структурування, залежностей та можливих проблем.

Першим кроком при EDA є збір інформації про дані. Наскільки вони великі, які вони мають типи даних, чи присутні в них пропущені значення, яка розподіленість даних і т.д. Ця інформація допомагає зрозуміти, які можливості наявні у ваших даних і як їх краще аналізувати.

Сюди входить також візуалізація даних, яка може допомогти виявити кореляції між змінними та можливі виклики, пов'язані з даними. Найпоширенішими інструментами для візуалізації є Python (мова програмування) бібліотеки *matplotlib* та *seaborn*, за допомогою яких ми можемо будувати різні лінійні графіки, діаграми, гістограми та інші.

Створення моделі – найскладніша, проте найцікавіша, частина роботи Data Scient'иста. Це етап, де формується гіпотеза і, саме тут відбувається магія, оскільки завдяки нашим даним (історичним) ми можемо «вчитись» розрізняти спам, передбачати ціну на будинок чи розблоковувати телефон дотиком чи обличчям.

Ще раз, перш, ніж перейти до цього етапу, майте на увазі, що етап очищення та дослідження даних не менш важливий для створення корисних моделей.

Тому поспіх із застосуванням моделювання не гарантує успіху, краще звернути увагу на ці етапи.

У моделюванні є кілька завдань. Ми також можемо навчати моделі виконувати класифікацію для розрізнення електронних листів, які отримує користувач як «Вхідні» та «Спам», використовуючи логістичну регресію.Також можемо прогнозувати значення, використовуючи лінійну регресію, або можемо використовувати моделювання для групування даних, щоб зрозуміти логіку за цими кластерами. Наприклад, ми групуємо клієнтів електронної комерції, щоб зрозуміти їх поведінку на вашому веб-сайті. Це вимагає від нас ідентифікувати групи точок даних з алгоритмами кластеризації, такими як k-середніх або ієрархічна кластеризація.

Для реалізації даного етапу нам вже потрібно знати бібліотеку *scikit-learn* а також бібліотеки для глибокого навчання такі як *TensorFlow* чи *PyTorch*, у разі використання великої к-сті даних та складніших моделей. А також ми повинні вміти робити оцінку роботи нашої моделі, щоб розуміти, яка модель краща, як можна покращити її роботу і т.д. Відповідно для оцінки моделі нам потрібні деякі метрики, як наприклад RMSE, MAE чи F1 score.

Інтерпретація (Застосування) – останній етапі життєвого циклу проєкту Data Science. Вся суть моделі в її прогностичній силі. Як модель «вміє» узагальнювати. Те, як пояснити модель, залежить від її здатності узагальнювати майбутні невідомі на теперішній момент дані.

Інтерпретація даних означає представлення ваших даних неспеціалісту в цій галузі. Ми демонструємо результати для відповіді на бізнес-питання, які ми ставили на початку проєкту, разом з дієвими інсайтами, які ми знайшли через власне Data Science, тобто наш процес дослідження даних. Це і є основна робота Data Scientist'a - принести користь компанії, зробивши якісь оптимізації чи діставши якісь інсайти з даних. До прикладу запропонувати той товар, котрий купить користувач, правильно застосувати таргетовану рекламу і т.д.

Дієвий інсайт є ключовим результатом, а отже ми вивчаємо, як повторити позитивний результат або запобігти негативному результату. Крім того, потрібно візуалізувати свої результати відповідно до вашого бізнес-питання. Важливо представити результати таким чином, щоб вони були корисні організації, інакше це буде безглуздо.

7.4 Алгоритми машинного навчання

На етапі побудови моделі, ми згадували деякі задачі та алгоритми, котрі існують для їх вирішення. Для кращого розуміння, які взагалі існують алгоритми, давайте зануримось у поточну класифікацію алгоритмів (рис. 7.2).

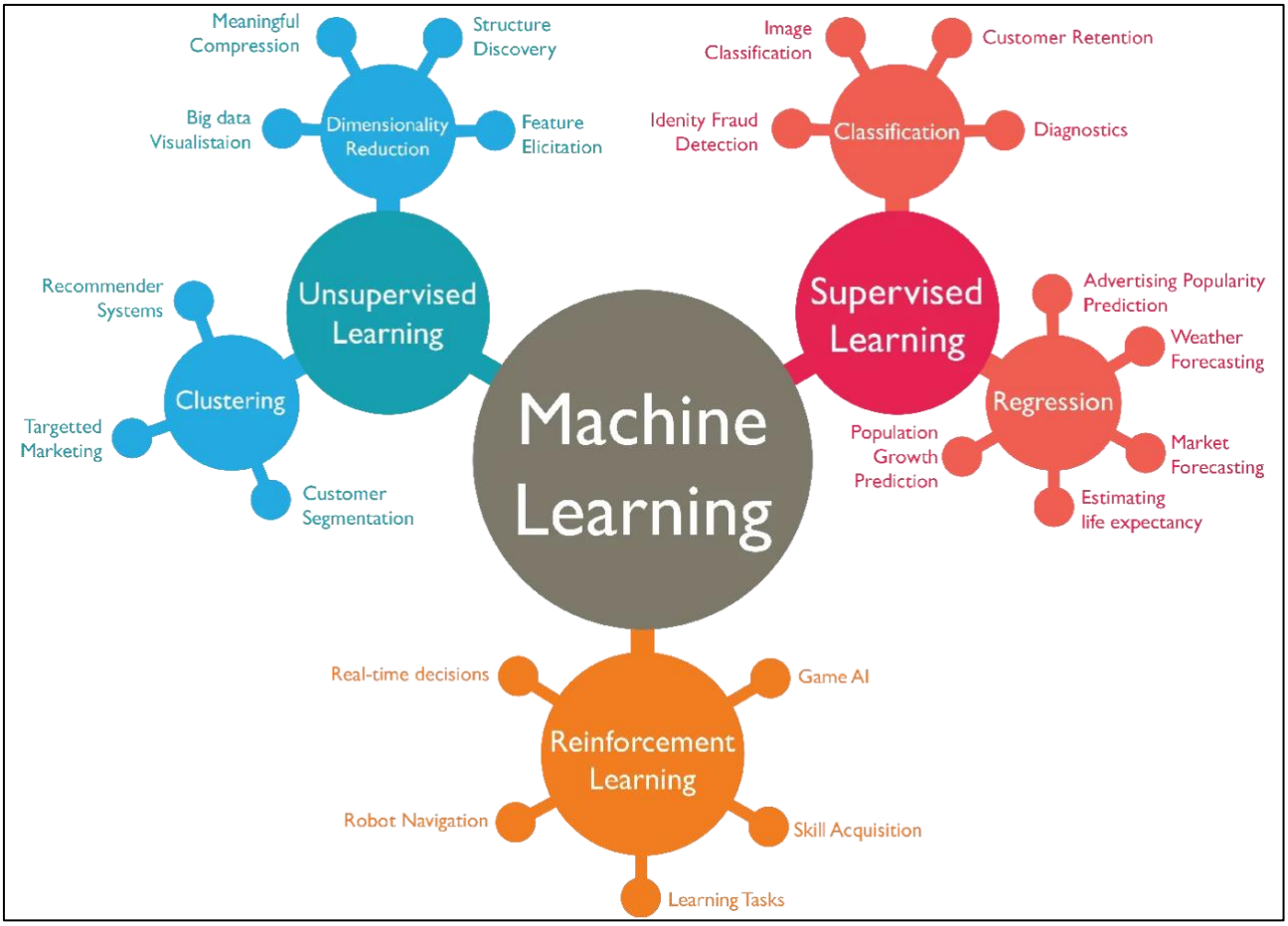

Рисунок 7.2 – Класифікація алгоритмів машинного навчання.

Навчання з учителем (англ. supervised learning) – це один з видів машинного навчання, в якому модель навчається на основі попередньо позначених (маркованих) даних. Іншими словами, для кожного набору вхідних даних є правильна відповідь, яку модель намагається передбачити. Модель тренується на цих позначених даних з метою знаходження патернів та створення правильних передбачень для нових, невідомих даних.

Один з прикладів застосування supervised learning – це задача класифікації електронної пошти на «спам» або «інші повідомлення». В даному випадку, модель будується на основі попередньо позначених даних, які містять інформацію про зразки спаму та нормальних повідомлень. Потім, модель навчається розпізнавати зразки та встановлювати, чи є нові повідомлення спамом чи ні.

Ще один приклад – це прогнозування цін на нерухомість. Модель може бути навчена на попередньо позначених даних, які містять інформацію про характеристики, ознаки нерухомості та їх вартість. Потім, модель навчається розпізнавати зв'язок між характеристиками та їх ціною, щоб передбачити ціну для іншої нерухомості (будинків, квартир, тощо).

В обох прикладах, модель тренується на попередньо позначених даних з метою передбачення правильних відповідей для нових, невідомих даних. Навчання з учителем дозволяє створювати точні прогнози та класифікації з

високим рівнем достовірності, що робить його корисним для широкого спектру задач у різних галузях.

Навчання без учителя (англ. unsupervised learning) – це один з підходів машинного навчання, який дозволяє виявляти приховані залежності в даних, без використання міток або результатів. Основна мета полягає у знаходженні структури в даних та кластеризації, де алгоритми виявляють групи схожих елементів у наборі даних, що не мають явних міток.

Наприклад, розглянемо задачу кластеризації, де маємо набір зображень та бажаємо розділити їх на різні групи. Натомість, у наборі даних немає інформації про те, який об'єкт до якої групи належить. Алгоритми навчання без вчителя, такі як *k-means*, можуть бути використані для кластеризації зображень на основі їхньої схожості.

Іншим прикладом є зменшення розмірності даних, де маємо великий набір даних із багатьма функціями, та хочемо зменшити кількість ознак для подальшого аналізу. У такому випадку, можна використовувати алгоритми пониження розмірності, такі як *Principal Component Analysis (PCA)*, для зменшення кількості ознак із збереженням якомога більше інформації.

Узагальнюючи, навчання без учителя – це підхід до машинного навчання, який дозволяє знаходити структуру в даних без попереднього навчання на вибірці даних. Це відкриває можливості для виявлення нових залежностей та патернів, які можуть бути корисними в багатьох областях, таких як рекомендації, обробка природних мов, аналіз даних та багато інших.

Рекомендаційні системи – це програмні рішення, що аналізують дані про користувачів та предмети, що їх цікавлять, з метою надання персоналізованих рекомендацій користувачам. Такі системи використовуються в різних сферах, включаючи електронну комерцію, засоби масової інформації, музику, соціальні мережі тощо.

У рекомендаційних системах використовуються два основних підходи: колаборативний та контент-базований. Колаборативний підхід базується на історії взаємодії користувача з системою та аналізу спільності цих взаємодій з іншими користувачами. Контент-базований підхід використовує відомості про властивості та характеристики предметів для порівняння з вподобаннями користувача.

Один з прикладів рекомендаційної системи – Netflix. Система аналізує історію перегляду користувача, їх вподобання та рейтинги фільмів, а також аналізує дані про характеристики самого фільму, такі як жанр, акторський склад, режисер тощо. На основі цих даних система робить персоналізовані рекомендації користувачам.

Інший приклад – Amazon. Рекомендаційна система Amazon аналізує історію покупок користувачів, їх відгуки та рейтинги товарів, які вони придбали. Система також аналізує властивості товарів, щоб порівняти їх з

вподобаннями користувача та знайти товари, які він може бути зацікавлений купити.

Рекомендаційні системи є важливим інструментом для підвищення рівня задоволення користувачів і збільшення прибутку компанії. Їх використовують в різних галузях, таких як електронна комерція, засоби масової інформації, музика, ігри та інші.

Навчання з підкріпленням (reinforcement learning) – є однією з головних підгалузей машинного навчання, яка полягає у тому, щоб навчити агента взаємодіяти з довкіллям, максимізуючи нагороду або мінімізуючи покарання.

У навчанні з підкріпленням немає правильних або неправильних відповідей. Замість цього, агент знаходиться у взаємодії з довкіллям і робить певні дії, за які отримує нагороду або покарання. Метою агента є вивчення стратегії, яка дозволяє йому максимізувати нагороду на довгострокову перспективу.

Один з прикладів застосування навчання з підкріпленням - гра в Го. Агент (наприклад, комп'ютерна програма) знаходиться взаємодії з довкіллям (Го-дошка) і має робити певні ходи, за які отримує очки, що відображають його успішність. Метою агента є навчитися грати в Го краще, щоб заробляти якомога більше очок. AlphaGo компанії Google навчилася грати в гру Го та перемогла одного з найкращих гравців у світі.

Інший приклад застосування навчання з підкріпленням – роботи, які виконують різні завдання. Робот взаємодіє з довкіллям, роблячи певні дії, і отримує нагороду або покарання в залежності від того, наскільки ефективно він виконує завдання.

Навчання з підкріпленням є досить складним процесом, оскільки агент має вирішувати проблеми з довгостроковою перспективою, де кожен крок може впливати на подальшу поведінку. Однак це дає можливість розв'язувати складні завдання, для яких немає простого рішення або алгоритму.

Нейронні мережі або глибоке навчання (англ. Neural networks or Deep Learning – це підгалузь машинного навчання, яка використовує нейронні мережі з багатьма шарами для автоматичного виконання завдань, таких як класифікація зображень, розпізнавання мовлення та обробка природної мови. Глибоке навчання базується на технології нейронних мереж, яка вивчає абстрактні характеристики даних, розпізнаючи взаємозв'язки між вхідними даними та вихідними мітками. Воно широко використовується в багатьох галузях, таких як медицина, фінанси та інші.

Залежно від того з якими даними ми працюємо, у нас є різні специфічні нейронні мережі, котрі більш ефективно використовувати у даному завданні.

Згорткові нейронні мережі (англ. Convolutional Neural Networks або CNN) є видом алгоритмів глибокого навчання, який спеціалізується на обробці даних, що мають сітчасту структуру, такі як зображення, відео та звук. Вони

використовуються для автоматичної обробки та аналізу великої кількості даних, що дозволяє їм виконувати завдання, такі як розпізнавання об'єктів на зображеннях, класифікацію зображень та інші пов'язані з цим задачі.

Основна ідея згорткових нейронних мереж полягає в тому, що вони використовують невеликі фільтри для згортки зображення та отримання відповідей на кожну з цих згорток. Далі, за допомогою шару підвибірки (pooling), отримані відповіді зменшуються, що дозволяє зменшити кількість параметрів та спростити обчислення. Застосування цих операцій дозволяє згортковим нейронним мережам розпізнавати об'єкти на зображеннях з високою точністю та ефективністю.

Прикладами застосування згорткових нейронних мереж є класифікація зображень, розпізнавання облич, відстеження руху та розпізнавання мови. Зокрема, вони широко використовуються в комп'ютерному зорі, рекомендаційних системах, робототехніці та інших областях, що пов'язані з обробкою даних зі зображень та відео.

Рекурентні нейронні мережі (RNN) – це тип нейронних мереж, який може аналізувати послідовності даних, де кожен елемент взаємозв'язаний з попередніми елементами послідовності (див. рис. 7.3).

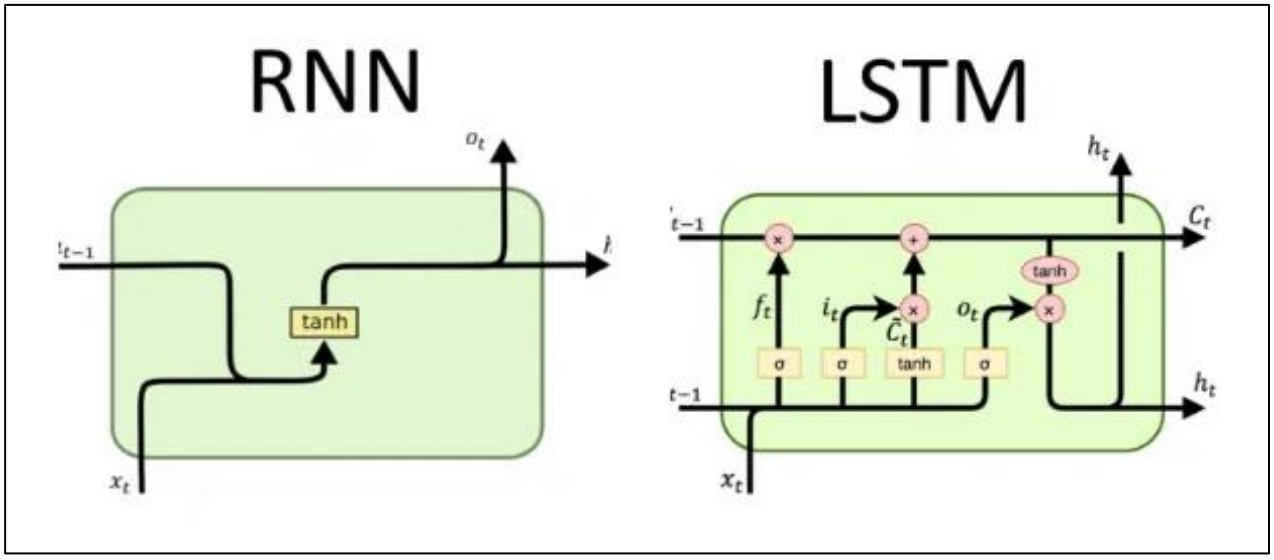

Рисунок 7.3 – Схеми рекурентних нейронних мереж.

Одна з основних особливостей RNN полягає в тому, що вони зберігають стан (пам'ять) з попередніх елементів вхідних даних і використовують його для здійснення прогнозів для наступного елементу. Це дозволяє їм враховувати контекст, що допомагає у більш точному прогнозуванні.

RNN зазвичай використовують для роботи з послідовностями даних, такими як мова, часові ряди, музика. Вони знаходять застосування в різних областях, включаючи машинний переклад, розпізнавання мови, генерацію тексту, прогнозування часових рядів та інше.

Один з найбільш відомих прикладів RNN – це *Long Short-Term Memory (LSTM)*, який є спеціальним типом RNN. Він може довгостроково

запам'ятовувати інформацію та уникнути проблеми зниклих градієнтів, що дозволяє йому бути дуже ефективним у роботі з послідовними даними. LSTM використовується для різноманітних завдань, включаючи автокомплітацію тексту, стиснення тексту, генерацію музики та багато іншого.

СПИСОК ЛІТЕРАТУРИ

1. Барабанова В.В., Богатирьова Г.А. Інноваційний маркетинг: навч. посібник. Кривий Ріг : Вид. ДонНУЕТ, 2022. 145 с. https://duikt.edu.ua/uploads/l_1390_85289741.pdf
2. Буга, Н., & Пелехацький, Д. (2022). Перспективи використання інноваційних технологій в маркетинговій діяльності підприємства. Економіка та суспільство, (42). https://doi.org/10.32782/2524-0072/2022-42-36
3. Виноградова О. В. Сучасні види маркетингу. Навчальний посібник. – Київ: ДУТ, 2019. 262 с. http://www.dut.edu.ua/ua/lib/1/category/743/view/1703
4. Грицунов О. В. Інформаційні системи та технології. Навчальний посібник. — Х.: ХНАМГ, 2010. — 222 с.
5. Джейсон Фрайд, Дэвид Хайнемайер Хэнссон. Rework. КСД. 2019. 176 с.
6. Джеф Сазерленд. Scrum. Навчись робити вдвічі більше за менший час. КСД. 2022. 280 с.
7. ДСТУ 2392-94. Інформація та документація. Базові поняття. Чинний від 1995-01-01. Вид. офіц. URL: https://dbn.at.ua/_ld/11/1166_DSTU2392-94.pdf (дата звернення: 22.07.2025).
8. ДСТУ ISO 21500:2022 (ISO 21500:2021) Управління проектами, програмами та портфелями. Контекст та концепції.
9. ДСТУ ISO 21502:2022 (ISO 21502:2020) Управління проектами, програмами та портфелями. Настанови щодо управління проектами.
10. ДСТУ ISO 21503:2022 (ISO 21503:2022) Управління проектами, програмами та портфелями. Настанови щодо управління програмами.
11. ДСТУ ISO 21504:2022 (ISO 21504:2022) Управління проектами, програмами та портфелями. Настанови щодо управління портфелями.
12. ДСТУ ISO 21505:2022 (ISO 21505:2017) Управління проектами, програмами та портфелями. Настанови щодо врядування.
13. ДСТУ ISO 21508:2022 (ISO 21508:2018) Управління здобутою цінністю в управлінні проектами та програмами.
14. ДСТУ ISO 21511:2022 (ISO 21511:2018) Ієрархічна структура робіт для управління проектами та програмами.
15. Ілляшенко С. М., Рудь М. П. Новітні види маркетингу в умовах випереджаючого розвитку: еволюція, сутність, умови застосування. Науковий вісник Ужгородського національного університету. Вип.24, ч.2. 2019. С.37-42. http://www.visnyk-econom.uzhnu.uz.ua/archive/24_2_2019ua/9.pdf
16. Кай-Фу Лі. Наддержави штучного інтелекту. BookChef. 2020. 240.

17. Кеті Кінг. Штучний інтелект у маркетингу. Як використовувати ШІ та бути на крок попереду. 2024. 257 с.
18. Курбан О.В., Курбан С.О. Нейромаркетинг: реклама, PR, digital-marketing, брендинг . Київ : Видавництво «Білий Тигр», 2019. 148 с. https://duikt.edu.ua/uploads/l_1515_51073371.pdf
19. Метелюк А. В., Чичур А. І. ПЕРСПЕКТИВИ РОЗВИТКУ BUSINESS INTELLIGENCE. Сучасний стан та перспективи розвитку IoT: Матеріали Всеукр. наук.-техн. конф., м. Київ, 15 квіт. 2025 р. / ДУІКТ, Каф. інформаційних систем та технологій. Київ, 2025. С. 89 - 91. Режим доступу: (https://duikt.edu.ua/uploads/p_2779_40288420.pdf).
20. Методи та засоби мультимедійних інформаційних систем: навч. посіб. / Т. М. Басюк, П. І. Жежнич; Нац. ун-т «Львів. політехніка». — Львів: Вид-во Львів. політехніки, 2015. — 426 с.
21. Петрина В. Інтелектуальний аналіз даних в якості методу обробки великих даних у сфері електронної комерції. Education and science of today: intersectoral issues and development of sciences. 2024. URL: https://doi.org/10.36074/logos-29.03.2024.069 (дата звернення: 14.03.2025).
22. Повалій Т. Л., Світайло Н. Д Івент-менеджмент : навчальний посібник. Суми: Сумський державний університет, 2021. 198 с. https://duikt.edu.ua/uploads/l_1413_31671579.pdf
23. Полянська А. С., Дюк О. М. Інтелектуальний аналіз даних у дослідженні корпоративної культури підприємства. Економічний вісник Національного технічного університету України «Київський політехнічний інститут». 2021. № 19. URL: https://doi.org/10.20535/2307-5651.19.2021.240691 (дата звернення: 14.03.2025).
24. Радіонова О. М Івент-технології : навч. посібник. Харків : ХНУМГ ім. О.М.Бекетова, 2021. 168с. https://duikt.edu.ua/uploads/l_1422_35934437.pdf
25. Соколов В. Ю. Інформаційні системи і технології : навч. посіб. Київ : ДУІКТ, 2010. 138 с. URL: https://duikt.edu.ua/uploads/l_603_15334144.pdf (дата звернення: 22.07.2025).
26. Anderson D. Kanban: Successful Evolutionary Change for Your Technology Business. Sequim, Washington : Blue hole press, 2010. 261 c.
27. Business intelligence in organizational decision-making: a bibliometric analysis of research trends and gaps (2014–2024) / S. Mekimah et al. Discover sustainability. 2024. Vol. 5, no. 1. URL: https://doi.org/10.1007/s43621-024-00692-7 (date of access: 14.03.2025).
28. Fabio Staiano. Designing and Prototyping Interfaces with Figma: Learn essential UX/UI design principles by creating interactive prototypes for mobile, tablet, and desktop. Packt Publishing. 2022. 382p.

29. Liang T.-P., Liu Y.-H. Research landscape of business intelligence and big data analytics: a bibliometrics study. Expert systems with applications. 2018. Vol. 111. P. 2–10. URL: https://doi.org/10.1016/j.eswa.2018.05.018 (date of access: 14.03.2025).
30. Project Managment Institute Ukraine. https://pmiukraine.org/
31. Project Managment Institute. https://www.pmi.org/.
32. Sean Adams (Шон Адамс). How Design Makes Us Think and Feel and Do Things. - ArtHuss. 2022. 256 с.
33. Stair R., Reynolds G. Principles of Information Systems - Fourteenth Edition. Cengage. USA. 2021. 584 p.
34. Tracy Osborne (Трейсі Осборн). Hello Web Design. Design Fundamentals and Shortcuts for Non-Designers. - Print2print. 2022. 176 с.
35. Xu J., Quaddus M. Managing information systems: ten essential topics. Atlantis Press (Zeger Karssen), 2013. 166 p.