ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

КВАЛІФІКАЦІЙНА РОБОТА

Тема: «УПРАВЛІННЯ ПРОЦЕСАМИ АВТОМАТИЗАЦІЇ ВИБОРУ МЕТОДОЛОГІЙ ДЛЯ ІТ-ПРОЄКТІВ»

Ступінь вищої освіти – магістр

Спеціальність – 073 «Менеджмент»

Освітня програма «Agile-технології розробки програмного забезпечення»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Керівник: д.е.н., доц., професор
кафедри ІММС
Ольга ОРЛОВА-КУРИЛОВА

Керівник: к. фіз-мат. н., доц.,
доцент кафедри ІММС
Іван КРИКУН

Виконав: здобувач
групи МЕН/Agile-23м
Євгеній ІВАЩЕНКО

Київ, 2024 р.

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

ЗАТВЕРДЖУЮ:
завідувач кафедри інформаційного менеджменту, математики та статистики
_____Денис БАЛДИК
«__»_____20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
ІВАЩЕНКО ЄВГЕНІЙ СЕРГІЙОВИЧ

Тема роботи	«Управління процесами автоматизації вибору методологій для ІТ-проєктів»
Номер та дата наказу про затвердження теми	№ 56-5 від 27 червня 2024 р.
Коротка постановка завдання	Розглянути завдання проєктного менеджменту шляхом застосування нейронних мереж. Провести аналіз моделі побудови нейронної мережі для аналізу проєкту. Провести аналіз датасету вхідних даних проєктів для нейронної мережі.
Посилання на джерела інформації (не більше п'яти найменувань, які рекомендує науковий керівник)	1. Agile Project Management: Best Practices and Methodologies [Електронний ресурс] // AltexSoft. – 2018. – Режим доступу до ресурсу: https://www.altexsoft.com/whitepapers/agile-project-management-best-practicesand-methodologies/ 2. What Is Scrumban? How It Differs from Scrum & Kanban [Електронний ресурс] // ProjectManager. – 2020. – Режим доступу до ресурсу: https://www.projectmanager.com/blog/what-is-scrumban. 3. What is a random forest? – Режим доступу до ресурсу: https://www.spotfire.com/glossary/what-is-a-random-forest
Вимоги до кваліфікаційної роботи	Кваліфікаційна робота має містити теоретичне та/або практичне дослідження за темою роботи, яку слід розглядати як складне спеціалізоване завдання або практичну проблематику в галузі управління та адміністрування, яка характеризується комплексністю та невизначеністю умов і потребує застосування теорій і методів Agile технологій.

Дата видачі завдання «14» липня 2024 р.

Керівник Ольга ОРЛОВА-КУРИЛОВА

Керівник Іван КРИКУН

Здобувач Євгеній ІВАЩЕНКО

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання	Примітка
Підготовчий етап			
1	Вибір напряму дослідження та керівника	01.07.2024 р.	Виконано
2	Формування теми та призначення керівника	08.07.2024 р.	Виконано
3	Затвердження теми кваліфікаційної роботи	09.07.2024 р.	Виконано
4	Затвердження завдання на кваліфікаційну роботу	15.07.2024 р.	Виконано
Основний етап			
5	Розробка концепції кваліфікаційної роботи	22.07.2024 р.	Виконано
6	Підбір та вивчення джерел інформації з напряму дослідження. Огляд існуючих аналогів.	29.07.2024 р.	Виконано
7	Затвердження розширеної постановки завдання. Підготовка та подання керівнику розділу 1 кваліфікаційної роботи	18.09.2024 р.	Виконано
8	Проєктування інформаційної системи. Підготовка та подання керівнику розділу 2 кваліфікаційноїроботи	18.09.2024 р.	Виконано
9	Реалізація інформаційної системи. Підготовка та поданнякерівнику розділу 3 кваліфікаційної роботи	25.09.2024 р.	Виконано
10	Підготовка та подання керівнику першого варіанту всієїкваліфікаційної роботи	01.10.2024 р.	Виконано
11	Доопрацювання кваліфікаційної роботи з урахуванням зауважень керівника та представлення керівнику доопрацьованого варіанту кваліфікаційної роботи	04.10.2024 р.	Виконано
Завершальний етап			
12	Представлення рукопису для перевірки на плагіат	07.10.2024 р.	Виконано
13	Підготовка презентації та доповіді на передзахист	07.10.2024 р.	Виконано
14	Передзахист кваліфікаційної роботи	08-11.10.2024 р.	Виконано
15	Технічна самоекспертиза роботи на відповідність вимогам до оформлення та виправлення недоліків	08-11.10.2024 р.	Виконано
16	Експертиза роботи керівником та зовнішнім експертом	14.10.2024 р.	Виконано
17	Доопрацювання доповіді та презентації для захисту	18.10.2024 р.	Виконано
18	Захист кваліфікаційної роботи	21-25.10.2024 р.	Виконано

Науковий керівник Ольга ОРЛОВА-КУРИЛОВА

Керівник Іван КРИКУН

Здобувач Євгеній ІВАЩЕНКО

АНОТАЦІЯ

Іващенко Є. С. «УПРАВЛІННЯ ПРОЦЕСАМИ АВТОМАТИЗАЦІЇ ВИБОРУ МЕТОДОЛОГІЙ ДЛЯ ІТ-ПРОЄКТІВ»

В кваліфікаційній роботі розглянуто теоретико-методичні основи вирішення проблем вибору методологій для ІТ-проєктів. У роботі розглянуто підходи до управління процесами автоматизації, включаючи аналіз ключових аспектів та вимог, які впливають на вибір методології, такі як розмір проєкту, команда, строк реалізації та специфіка завдань. Окрема увага приділяється використанню сучасних технологій, таких як нейронні мережі та алгоритми машинного навчання, для прогнозування оптимальної методології залежно від параметрів проєкту.

Робота досліджує різні методології, включаючи Agile, Waterfall, Kanban та інші, з точки зору їх застосовності у різних типах ІТ-проєктів. Описані алгоритми та технічні рішення дозволяють автоматизувати процес вибору та підвищити ефективність управління проєктами.

Основна мета полягає у створенні автоматизованої системи, яка допоможе ІТ-менеджерам швидко й точно вибирати відповідну методологію, враховуючи характеристики проєкту, ресурси та очікувані результати.

Ключові слова: Проєктний менеджмент, автоматизація, Agile, Waterfall, Kanban, нейронні мережі, машинне навчання, методологія, гнучке управління.

ANNOTATION

Ivashchenko I. S. " MANAGEMENT OF AUTOMATION PROCESSES FOR SELECTING METHODOLOGIES FOR IT PROJECTS"

This qualification work is dedicated to solving the problem of automated methodology selection for IT projects. It reviews approaches to managing automation processes, including the analysis of key aspects and requirements that influence the choice of methodology, such as project size, team composition, timelines, and the specifics of tasks. Special attention is given to the use of modern

technologies, such as neural networks and machine learning algorithms, to predict the optimal methodology based on project parameters.

The work explores various methodologies, including Agile, Waterfall, Kanban, and others, in terms of their applicability to different types of IT projects. The described algorithms and technical solutions allow for the automation of the selection process and improve project management efficiency.

The main goal is to create an automated system that helps IT managers quickly and accurately choose the appropriate methodology, taking into account project characteristics, resources, and expected outcomes.

Key words: Project management, automation, Agile, Waterfall, Kanban, neural networks, machine learning, methodology, agile management.

ЗМІСТ

ВСТУП 7

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ ... 8

1.1 Особливості проєктного менеджменту у ІТ-галузі 8

1.2 Запровадження методологій проєктного менеджменту і експертні оцінки 17

1.3 Проблема вибору методології ІТ-проєкту за заданими умовами 23

Висновки до розділу 1 40

РОЗДІЛ 2 ІНФОРМАЦІЙНИЙ АНАЛІЗ ПРОЄКТНОЇ ДІЯЛЬНОСТІ ПІДПРИЄМСТВА 41

2.1 Вирішення завдань проєктного менеджменту шляхом застосування нейронних мереж 41

2.2 Аналіз моделі побудови нейронної мережі для аналізу проєкту 44

2.3 Аналіз датасету вхідних даних проєктів для нейронної мережі 48

Висновки до розділу 2 50

РОЗДІЛ 3 ПОБУДОВА МОДЕЛІ ДЛЯ АВТОМАТИЗАЦІЇ ВПРОВАДЖЕННЯ МЕТОДОЛОГІЇ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ 51

3.1 Математичні основи побудови моделі автоматизації впровадження методології 51

3.2 Постановка гіпотези щодо впливу показників на вибір методології проєктного менеджменту у ІТ-проєктах 55

3.3 Оцінка ефективності отриманої моделі 57

Висновки до розділу 3 63

ВИСНОВКИ 64

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 66

ВСТУП

Проєктна діяльність супроводжується численними ризиками та впливом факторів, що ускладнюють її прогнозування. ІТ-проєкти мають особливості, які вирізняють їх серед інших, зокрема стрімкий розвиток технологій та постійні зміни на ринку. Успіх таких проєктів залежить від багатьох чинників: вимог замовника чи інвестора, динамічних змін у вимогах, технічного забезпечення та внутрішньої взаємодії команди. Ефективне управління цими чинниками можливе завдяки правильно обраній методології проєктного менеджменту, яка прямо впливає на результативність роботи команди.

Вибір методології управління проєктами є відповідальністю проєктного менеджера, проте це рішення часто приймається колективно: операційним директором, проєктним менеджером та власником продукту. Процедури вибору чи зміни методології можуть бути регламентовані бізнес-процесами компанії та потребувати бюрократичних погоджень.

Оскільки процес вибору методології є досить типовим і повторюваним, його можна оптимізувати шляхом автоматизації. Для цього необхідно визначити ключові фактори успіху проєкту та проаналізувати їх вплив на вибір відповідної методології.

Об’єкт дослідження – процеси впровадження методологій управління ІТ-проєктами.

Предмет дослідження – використання нейронних мереж для вирішення задачі вибору методології управління проєктами у компанії.

Наукова новизна – запропоновано методику застосування нейронних мереж у проєктній діяльності, що дозволяє прогнозувати найоптимальнішу методологію проєктного менеджменту на етапі ініціації проєкту.

Такий підхід сприятиме підвищенню ефективності управління проєктами в умовах сучасних ІТ-компаній.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ

1.1 Особливості проєктного менеджменту у ІТ-галузі

Зростаючий попит на розробку програмного забезпечення в Україні актуалізує потребу в структурованому управлінні ІТ-проєктами. Оскільки кожне підприємство має свої унікальні процеси та цінності, впровадження систем управління проєктами відбувається індивідуально. Водночас, універсальна концепція SDLC визначає загальні етапи розробки програмних продуктів, впливаючи на організацію цього процесу в різних компаніях.

Правильна класифікація ІТ-проєкту є ключовим фактором для успішної реалізації. Різні типи проєктів вимагають різних підходів до управління. Розглянемо, як класифікація впливає на вибір методології.

Класифікувати проєкти можна за ознаками показаними на Рис. 1.1 [1], а також відповідно до Таблиці 1.1.

1. Масштаб проєкту, поділяється на:

   a. малі проєкти: Часто використовують гнучкі методології (Agile, Scrum), оскільки вони дозволяють швидко адаптуватися до змін та зосередитися на найбільш важливих функціоналах.

   b. великі проєкти: Частіше застосовують традиційні методології (Waterfall), які забезпечують детальне планування та контроль над усіма етапами проєкту.

2. Складність проєкту, поділяється на:

   a. прості проєкти: Можуть бути реалізовані за допомогою простих методологій або навіть без формального управління проєктами.

   b. складні проєкти: Вимагають використання комплексних методологій, які враховують велику кількість змінних та взаємодій.

3. Тривалість проєкту, поділяється на:

   a. короткострокові проєкти: Часто використовують ітеративні методології, які дозволяють швидко отримати результат.

b. довгострокові проєкти: Вимагають більш структурованих підходів, які забезпечують стабільність та прогнозованість.

4. Тип проєкту, поділяється на:

a. Інноваційні проєкти: Часто використовують гнучкі методології, які дозволяють швидко адаптуватися до нових ідей та технологій.

b. Проєкти з чітко визначеними вимогами: Можуть бути реалізовані за допомогою традиційних методологій.

Таблиця 1.1 Відповідності класифікації та методологій

Класифікація проєкту	*Характеристики*	*Типові методології*
Малий, простий, короткостроковий	Невеликий обсяг роботи, чітко визначені вимоги	Agile, Scrum, Kanban
Великий, складний, довгостроковий	Великий обсяг роботи, багато учасників, високий рівень невизначеності	Waterfall, PRINCE2
Інноваційний	Нові технології, швидкі зміни вимог	Agile, Scrum, DevOps

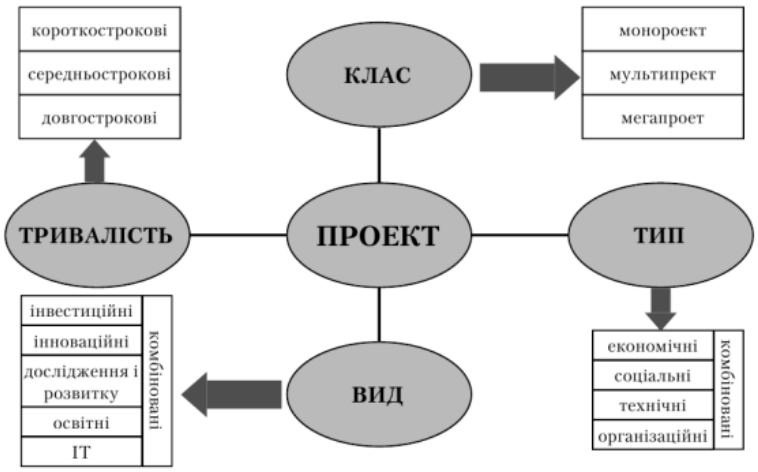


Рисунок 1.1 – Класифікація проєктів за основними ознаками [1]

Багаторівнева структура офісу проєктного менеджменту дозволяє ефективно управляти великими та складними проєктами, особливо в організаціях з розподіленою структурою. Кожен рівень відповідає за певні аспекти управління проєктами:

1. Рівень проєкту.
2. Рівень програми.
3. Рівень портфеля.
4. Корпоративний рівень.

Впровадження корпоративної інформаційної системи (КІС) є важливим кроком для підвищення ефективності управління проєктами. КІС дозволяє наступне:

- об'єднати всі рівні управління проєктами: Забезпечити єдину точку доступу до інформації про всі проєкти, програми та портфелі;
- автоматизувати рутинні процеси: Звільнити співробітників від виконання рутинних задач і дозволити їм зосередитися на стратегічних питаннях;
- підвищити прозорість: Забезпечити доступ до інформації про проєкти для всіх зацікавлених сторін;
- поліпшити прийняття рішень: Надати менеджерам актуальну і достовірну інформацію для прийняття обґрунтованих рішень;

Для успішного впровадження КІС необхідно виконати наступні дії:

- залучити всіх зацікавлених сторін: Менеджерів проєктів, програми, портфеля, а також бізнес-користувачів;
- розробити детальний план впровадження: Визначити цілі, вибрати систему, розробити процес міграції даних;
- забезпечити підготовку користувачів: Провести навчання і забезпечити технічну підтримку;
- поступово розгортати систему: Почати з пілотних проєктів і поступово розширювати охоплення;

Багаторівнева структура офісу проєктного менеджменту є ефективним інструментом для управління великими і складними проєктами. Впровадження корпоративної інформаційної системи дозволяє автоматизувати процеси, підвищити прозорість і прийняття рішень.

Кожна методологія управління проєктами має свої унікальні риси. Особливу увагу слід приділяти визначенню пріоритетів у кожній методології. Вибір підходу здебільшого залежить від потреб замовників: чи хочуть вони швидко отримати продукт із мінімальною кількістю помилок, чи віддають перевагу участі в процесі, спостерігаючи за кожним етапом. Різні методології відповідають різним потребам і пріоритетам.

У деяких методологіях пріоритети є очевидними, в інших — менш виражені [3]. Наприклад, об'єктно-орієнтована методологія Мартіна й Оделла є доволі універсальною, проте не завжди зрозуміло, на що вона орієнтована і чи підходить для різних типів проєктів. Методології OPEN зосереджуються на забезпеченні правильності програмних рішень, прозорості та повторюваності процесів. Методологія The Personal Software Process Гамфрі була розроблена для досягнення передбачуваності виконання завдань.

Методології Crystal та Extreme Programming (XP) чітко формулюють свої пріоритети: обидві зосереджені на підвищенні продуктивності й зниженні витрат. Проте вони мають суттєві відмінності: Crystal прагне поєднувати продуктивність із гнучкістю, тоді як XP збільшує продуктивність шляхом зменшення гнучкості. Методологія Adaptive Software Development Джима Хайсміта розроблена для роботи в умовах постійних змін вимог, проєктування та стислих термінів, що часто трапляється під час розробки веб-додатків.

Варто зазначити, що на методологію значною мірою впливає досвід її автора. Кожен елемент методології спрямований на уникнення проблем, з якими автор стикався раніше. Наприклад, менеджер проєкту може побоюватися, що програмісти допустять багато помилок у коді, тому він впроваджує регулярні перевірки. Інший менеджер може вважати, що замовники не до кінця визначилися з вимогами, тому акцент робиться на

створенні прототипів інтерфейсу. Якщо менеджер хвилюється, що розробники можуть піти з проєкту, він зосереджується на створенні детальної документації всіх етапів. Таким чином, методології часто відображають переживання і досвід їхніх творців.

Останній аспект, який формує методологію, — це індивідуальна філософія її автора. Ця філософія випливає з особистого досвіду і формується на основі певних висновків, зроблених із цього досвіду. Щоб методологія була ефективною для певного проєкту, вона повинна відповідати філософії як команди розробників, так і самого автора.

Існує багато різних методологій, які відрізняються кількістю учасників проєкту, рівнем критичності, пріоритетами та підходами до розробки. Вибір методології також залежить від поглядів спонсорів: що для них важливіше — продуктивність, прозорість, повторюваність чи точність процесу. «Масштаб» методології зростає зі збільшенням кількості учасників і комунікаційних процесів, а її «щільність» — зі збільшенням контролю [3].

Цей підхід відображає реальну ситуацію в проєктах. Ми можемо оцінити кількість учасників, обговорити критичність і пріоритети. Використовуючи ці принципи, можна прийняти рішення щодо вибору методології для конкретного проєкту.

Однією з основних проблем у проєктному менеджменті є необхідність змінювати методологію в процесі виконання робіт. Одним із ключових факторів є адаптація методології під час реалізації проєкту. Кожен проєкт потребує індивідуального підходу, тому вибір методології на початку є лише першим кроком до того, що може виявитися найкращим підходом.

Якщо регулярно отримувати зворотний зв’язок від команди між етапами проєкту, можна вчасно коригувати методологію. Це дасть можливість покращувати процес управління проєктом у реальному часі.

Отже, будь-яка методологія складається з таких основних компонентів: ролей, навичок, видів діяльності, технік, інструментів, артефактів, стандартів, критеріїв якості та пріоритетів проєкту. Залежно від розміру, критичності

кінцевого продукту та основних пріоритетів, можна використовувати різні методології. Творці методологій обирають певні аспекти (ролі, види діяльності, стандарти) і намагаються мінімізувати ризики, ґрунтуючись на власному досвіді, страхах та філософії. Під час вибору методології необхідно враховувати ці аспекти та їхню відповідність потребам проєкту чи організації.

Сучасні великі проєкти інформаційних систем мають такі характеристики [3]:

1. Складність опису: Велика кількість функцій, бізнес-процесів, інформаційних даних і складних взаємозв'язків між ними. Це питання вирішується шляхом моделювання та аналізу даних і процесів.

2. Наявність внутрішніх підсистем: Підсистеми тісно взаємодіють одна з одною, мають власні підцілі та підзадачі. Наприклад, одні системи обробляють транзакції та вирішують конкретні регламентовані завдання, тоді як інші підтримують прийняття рішень за допомогою нерегламентованих запитів до великих масивів даних.

3. Інтеграція модулів: Поєднання вже існуючих і нових, нещодавно розроблених модулів.

4. Функціонування в неоднорідному середовищі: Підтримка роботи на різних апаратних платформах або операційних системах.

5. Різнорідність кваліфікації команди: Участь фахівців із різними рівнями компетентності та володінням різними технологічними стеками під час розробки.

6. Динамічність та постійні зміни: Постійна еволюція структури й вимог до інформаційних систем.

Життєвий цикл проєкту визначає концептуальний підхід до організації робіт і, в багатьох випадках, основні фази цього циклу разом із принципами переходу між ними. Методологія задає комплекс завдань, деталізує зміст кожної фази та розподіляє відповідальність між фахівцями. Вона також визначає модель життєвого циклу і пропонує найкращі практики (best

practices), що допомагають максимально ефективно використовувати обрану методологію.

В IT-проєктах важливо не лише використовувати загальні принципи управління проєктами, а й конкретизувати фази проєкту, чітко їх розмежовуючи. Це не означає, що вони повинні виконуватися послідовно або лінійно.

Модель життєвого циклу — це структура, що включає процеси, завдання та роботи, пов'язані з розробкою, експлуатацією та підтримкою інформаційної системи або програмного продукту. Вона охоплює весь життєвий цикл системи — від постановки вимог до моменту її виведення з експлуатації.

Скотт Амблер, котрий є автором концепцій гнучкого моделювання (Agile Modeling) та Enterprise Unified Process (розширення Rational Unified Process), визначає такі рівні життєвого циклу, що зумовлені відповідними видами робіт (рис. 1.2) [3]:

- цикл життя розробки програмного забезпечення (SDLC): проєктна діяльність, пов'язана із створенням і впровадженням програмних систем;
- цикл життя програмної системи: охоплює етапи розробки, впровадження, підтримки та супроводу.
- цикл життя інформаційних технологій (IT): включає всі дії, які здійснює IT-департамент.
- цикл життя організації/бізнесу: охоплює всі аспекти діяльності організації в цілому.

Рисунок 1.2 – Зміст категорій життєвих циклів на підприємстві у галузі IT [3]

Для автоматизації впровадження методології управління IT-проєктами доцільно використовувати вже існуючі програмні продукти для управління проєктами, розширюючи їх функціональні можливості відповідно до специфічних бізнес-завдань. Щоб досягти цієї мети, необхідно виконати такі завдання [4]:

1. Переглянути поточну систему управління проєктами та контролю, щоб оцінити готовність компанії до змін і рівень її зрілості.
2. Визначити основні вимоги до нових функціональних блоків інформаційної системи управління проєктами та кроки їх впровадження.

Методології управління проєктами можна визначити як набір найкращих практик, інструментів і методів, які є динамічними, гнучкими й адаптивними, тобто такими, що можуть підлаштовуватися під різні проєкти в конкретному середовищі. Тому методологія повинна включати процеси, шаблони, прийоми та інструменти, що допомагають планувати та управляти проєктом протягом всього його життєвого циклу. Компоненти методології охоплюють процеси управління проєктами: ініціювання, планування, досягнення результатів та моніторинг прогресу проєкту, підбір інструментів і методів для обміну зворотнім зв’язком із усіма зацікавленими сторонами, а

також консолідований і інтегрований набір відповідних найкращих практик та цінностей управління проєктами, разом з термінологією для забезпечення спільної мови в проєктному середовищі.

При виборі методології управління проєктами важливо враховувати кілька ключових аспектів, пов’язаних із самим проєктом [5]:

1. Хронологія проєкту.
2. Складність проєкту.
3. Бюджет проєкту.
4. Потреба в ресурсах.
5. Залучення зацікавлених сторін.
6. Галузь.

Вибір методології проєктного менеджменту є доцільним за наступних умов [6]:

- обсяг проєкту чітко визначений;
- витрати на проєкт розраховані;
- потреби зацікавлених сторін виявлені;
- всі конфлікти та залежності ідентифіковані й усунені;
- усі задачі вирішуються за єдиною методикою та підходом.

Сьогодні найпоширеніші моделі розробки програмного забезпечення поділяються на каскадні та гнучкі, які представляють дві основні групи методологій. Проте спроби впровадити програмні продукти для оптимізації управління проєктами часто зводяться до простого використання окремих можливостей різних інформаційних систем. Це пояснюється складністю таких систем, їх багатозадачністю та універсальністю для різних галузей. Такі фактори ускладнюють і роблять дорогим підлаштування під специфічні процеси розробки.

Додатково, рівень технологічного розвитку компанії є важливим чинником. На практиці інтеграція системи управління проєктами зазвичай закінчується лише на етапі документування бізнес-процесів. Як результат, впроваджується інформаційна система, яка використовується виключно для

створення графіків процесу розробки з можливістю деталізації. Управління часом здійснюється в обмеженому форматі, а повноцінне планування та управління вартістю часто не враховується .

1.2 Запровадження методологій проєктного менеджменту і експертні оцінки

У стандарті ISO 21500:2012 [7], який розробив проєктний комітет ISO/PC 236 «Управління проєктами», зазначено, що проєкт складається з унікального набору процесів. Ці процеси включають координовані та контрольовані операції з визначеними датами початку та завершення, які виконуються для досягнення поставленої мети.

Успіх досягнення основної мети проєкту, а також послідовність виконання завдань і їх пріоритетність, безпосередньо залежать від вибраної методології. У цій роботі розглядаються сучасні широко використовувані методології та підходи до розробки програмного забезпечення, які можна узагальнити до двох основних конкуруючих фреймворків: каскадних і гнучких.

Класичний цикл розробки програмного забезпечення дозволяє упорядкувати процес розробки, створити план і часовий графік виконання робіт. Проте ця модель має суттєвий недолік: вона ґрунтується на припущенні про точність формулювання вимог до програмного забезпечення. У реальності на початку проєкту вимоги визначаються лише частково. Оскільки замовник отримує доступ до результатів проєкту тільки після завершення всіх робіт, він не має можливості уточнити вимоги і, як правило, отримує продукт, який не відповідає його очікуванням. Саме цей недолік робить класичний цикл розробки програмного забезпечення практично непридатним.

Практика показує наступне - замовники часто не можуть чітко визначити свої потреби щодо програмного продукту. Більш того, сам процес розробки може вплинути на внутрішні процеси компанії та змінити початкові вимоги.

Хоча перехід до інженерних методологій і каскадної моделі був прогресивним кроком, вони не змогли вирішити всі проблеми програмних проєктів. Ці методології, розроблені за зразком інших інженерних дисциплін, не повністю враховують унікальні особливості програмного забезпечення.

Один з основних конфліктів полягає в недооцінці людського фактора. Інженерні методології часто фокусуються на технічних аспектах, ігноруючи соціальні та людські фактори, які насправді є ключовими для успіху проєкту.

Інший конфлікт пов'язаний з невідповідністю каскадної моделі природі програмного забезпечення. Розробка програмного продукту є унікальним процесом з високою невизначеністю, і спроба передбачити всі фактори на початку проєкту часто закінчується невдачею.

Недоліки каскадної моделі не лише ускладнюють розробку, але й негативно впливають на людські відносини. Наприклад, детальне визначення вимог на початку проєкту може призвести до конфліктів між замовником і виконавцем при необхідності внесення змін.

Хоча каскадна модель може працювати для деяких проєктів, для багатьох сучасних продуктів її недоліки є актуальними і часто призводять до провалу проєктів.

Перехід до інженерних методологій, включно з каскадною моделлю, став важливим етапом у впорядкуванні процесу розробки програмного забезпечення та усуненні багатьох проблем підходу "розробляй і виправляй" (code-and-fix). Проте ці методології не могли вирішити всі проблеми програмних проєктів, оскільки містили внутрішній конфлікт. Вони були створені за аналогією з іншими інженерними дисциплінами, але недостатньо враховували специфіку розробки програмного забезпечення, яке має свої унікальні особливості.

Цей конфлікт проявляється двома основними шляхами :

Перший прояв проблеми - це недооцінка людського фактора: Інженерні методології орієнтувалися на виконання чітко визначених кроків, які розглядалися незалежно від людей, що їх виконують. Вони зосереджувалися

на процесах, а не на команді, яка безпосередньо займається розробкою. Однак, успіх проєкту значною мірою залежить від взаємодії учасників, їхніх відносин і командної роботи.

Другий прояв проблеми полягає в тому, що каскадний життєвий цикл не відповідає природі програмного забезпечення. Особливість програмних продуктів полягає в їхній унікальності та новизні, а процес розробки відбувається в умовах високої невизначеності. Спроба врахувати всі можливі фактори на початку проєкту (скласти детальні вимоги, створити дизайн системи тощо) приречена на невдачу. Ще гірше те, що недоліки каскадної моделі не тільки ускладнюють сам процес розробки, але й погіршують відносини між учасниками проєкту. Наприклад, затвердження детальних вимог на початковому етапі часто призводить до конфліктів між замовником і розробником, якщо виникає потреба внесення змін під час реалізації. Це викликає суперечки, погіршує відносини і руйнує ефективну співпрацю команди із замовником.

Варто зазначити, що є проєкти, для яких каскадна модель може працювати добре, але для багатьох сучасних програмних продуктів ці проблеми є надзвичайно актуальними, і часто призводять до провалу проєктів.

Інженерні методології орієнтовані на детальне планування. Задачі і ресурси визначаються на весь період проєкту, і команда важко адаптується до змін. Якщо змінюються вимоги, це може спричинити значні зміни в плані, термінах і бюджеті проєкту. Постійні невдачі проєктів, що використовували інженерні методології, стимулювали пошук альтернатив.

З середини 90-х років набули популярності нові підходи – "гнучкі методології". Це методи розробки програмного забезпечення, що засновані на ітеративній розробці, гнучкому зборі вимог та постійній взаємодії всередині самоорганізованих команд фахівців різного профілю.

У гнучких методологіях використовується інкрементна модель життєвого циклу, що розбиває процес розробки на окремі етапи (інкременти),

кожен з яких включає ключові фази каскадної моделі: аналіз, проєктування, програмування і тестування.

Гнучкі методології допомагають справлятися з неповнотою вимог і їх постійними змінами. Команда, яка працює за гнучкою методологією, зазвичай може передбачити лише найближчі завдання. Довгострокові плани залишаються орієнтовними і включають лише грубі оцінки витрат, часу і очікуваних результатів.

На таблиці 1.2 показані загальні спільні і відмінні риси груп Методологій [9].

Таблиця 1.2 Порівняння каскадних та гнучких груп методологій управління проєктами

ОЗНАКИ	*МЕТОДОЛОГІЇ*	
	ГНУЧКІ	*КАСКАДНІ*
ПОСЛІДОВНА РОЗРОБКА		+
ГНУЧКА РОЗРОБКА	+	
МОЖЛИВІСТЬ ЗМІН	+	
ВИЗНАЧЕНІ ВИМОГИ		+
ЗАБЕЗПЕЧЕННЯ ЯКІСНОГО РЕЗУЛЬТАТУ	+	+
ПОСТІЙНИЙ РОЗВИТОК	+	
РЕГЛАМЕНТОВАНІ ЕТАПИ РОЗРОБКИ		+

Виділяють кілька ключових принципів при виборі методології управління IT-проєктами [5]:

Принцип 1. Обсяг методології залежить від кількості розробників, залучених до проєкту. Чим масштабніший проєкт, тим більше учасників і ролей він включає. Відповідно, методологія повинна адаптуватися до цих

обставин. Це підкреслює важливість гнучкості під час вибору методології — те, що працює в одній ситуації, може не бути ефективним в іншій.

Принцип 2. Чим більшої точності вимагає проєкт або система, тим ретельніше має бути обрана методологія. Потенційні ризики (від незначних проблем до серйозних фінансових збитків або навіть загрози життю) визначають рівень захисту та передбачення помилок. Проте надмірно складна методологія може уповільнити процес розробки та збільшити витрати.

Принцип 3. Зміна масштабу чи складності методології безпосередньо впливає на бюджет. Чим більші та більш автономні команди, тим більше ресурсів потрібно для координації їхньої роботи. Оптимальними вважаються команди до 9-12 осіб, які працюють швидше та ефективніше. Однак масштабні проєкти потребують більших команд і "важкої" методології, що збільшує управлінську складову.

Принцип 4. Найбільш ефективна форма взаємодії команди — це безпосередня комунікація. Розробники, які мають можливість спілкуватися віч-на-віч, працюють ефективніше, що знижує витрати на розробку. Якщо ж проєкт розширюється і залучає віддалені команди, час виконання завдань зростає, що призводить до збільшення витрат.

Оскільки нас цікавить етап розробки, а інколи й підтримки ІТ-продукту в контексті проєкту, розглянемо систему показників, яку запропонувала Л. Л. Калініченко [10] для визначення ефективності командної роботи.

Ефективним управлінням можна вважати таке управління проєктною командою, яке дозволяє досягти найкращих результатів у найкоротші терміни. Виходячи з цього, Л. Л. Калініченко розробила формулу оцінки ефективності управління командою проєкту (Еукп), враховуючи час, витрачений на роботу над проєктом (Тукп):

$$Еукп = (A + B + C + D + E + F + G + H) / Тукп, \quad (1.1)$$

Де А – управління постановкою цілей, створення умов для того, щоб всі члени команди чітко розуміли мету проєкту;

B – досягнення мети командної роботи за такими критеріями: проєкт завершено вчасно і з запланованим прибутком; проєкт завершено вчасно, але результат не відповідає очікуванням; проєкт не завершено в рамках запланованого терміну;

C – формування командної взаємодії, що дозволяє згуртувати команду для подальшої роботи на інших етапах або нових проєктах;

D – управління делегуванням обов'язків, коли кожен член команди відповідальний за свій сегмент роботи;

E – управління командною роботою, сприяння ефективній співпраці та формування корпоративної причетності;

F – управління конкурентоспроможністю команди, сприяння розвитку професійних та особистих якостей членів команди;

G – управління гнучкістю та адаптивністю команди, яка може швидко переналаштуватися для роботи над іншим проєктом або в нових умовах;

H – управління безперервним вдосконаленням і розвитком компетенцій членів команди, коли після завершення проєкту вони набувають нових професійних і особистих навичок.

Оцінка зазначених вище компонентів управління командною роботою через економічні показники є складним завданням, оскільки ці показники мають конкретний фінансовий зміст і не завжди дозволяють точно оцінити вплив людського фактору. Тому найбільш ефективним способом оцінки управління командою, що займається проєктною діяльністю, вважається експертне оцінювання. Отже, визначити рівень розвитку управлінських компонентів проєктної команди доцільно за допомогою опитування експертів, яке включає оцінку вищезазначених показників. Щоб підвищити надійність і достовірність результатів експертних оцінок, важливо враховувати теоретичні й методологічні аспекти цих моделей.

Зазвичай рішення щодо впровадження методологій управління проєктами в окремому проєкті чи на підприємстві приймаються менеджером проєкту разом із операційним директором, з урахуванням існуючих процесів і

процедур компанії. Оскільки це стратегічне рішення, особливо для довготривалих проєктів, воно може ухвалюватися на вищому рівні управління. Процес прийняття рішення залежить від розміру організації, її корпоративної структури та бізнес-процесів. Якщо регламентовані процеси не диктують інше, рішення приймається на основі специфіки проєкту операційним директором (COO) разом із менеджером проєкту. Однак на вибір методології впливають численні фактори, зокрема ризики, які складно передбачити. Хоча експертна оцінка містить у собі елемент людського фактору, що підвищує ймовірність помилки, застосування алгоритмів машинного навчання може значно знизити ці ризики і зробити рішення більш точними

1.3 Проблема вибору методології IT-проєкту за заданими умовами

Окремі фази та стадії, що поєднують спільні риси IT-проєктів виділили у концепцію під назвою SDLC – Software Development Lifecycle або життєвий цикл розробки програмного забезпечення. Життєвий цикл розробки програмного забезпечення (SDLC) – це систематичний процес розробки програмного забезпечення, що забезпечує якість та правильність розробленого кінцевого продукту, спрямований на виробництво високоякісного програмного забезпечення, яке відповідає очікуванням клієнтів.

Важливим акцентом концепції є те, що розробка системи повинна бути завершеною за попередньо визначеними часовими рамками та витратами.

SDLC складається з детального гайдлайну, який пояснює, як планувати, будувати та підтримувати конкретне програмне забезпечення. Кожна фаза життєвого циклу SDLC має свій процес і результати, які переходять у наступну фазу.

Переваги SDLC для розробки інформаційної системи [11]:

- пропонує основу для ініціації проєктів, їх планування та оцінки;
- забезпечує основу для стандартного набору заходів та результатів;
- виступає механізмом відстеження та контролю виконання проєктів;

- підвищує наочність планування проєктів для всіх зацікавлених сторін;
- процесу розробки;
- збільшує продуктивність та підвищує швидкість розробки;
- поліпшує взаємодію проєктної команди з клієнтами;
- допомагає зменшити потенційні ризики та відхилення від початкового плану проєкту.

Існують чітко визначені етапи процесу SDLC. У деяких джерелах вони дещо видозмінені – їх більше або менше, вони об’єднані або замінені відповідно до специфіки розглянутих проєктів. Однак в загальному випадку існує 7 етапів, які будуть перелічені нижче [11].

Фаза 1: Збір та аналіз вимог. Узгодження вимог є першим етапом процесу SDLC. Збір та аналіз вимог проводить менеджер команди, бізнес аналітик та інші досвідчені члени команди за участю всіх зацікавлених сторін та експертів у галузі. На цьому етапі також проводиться планування внутрішніх вимог щодо забезпечення якості, критеріїв готовності продукту, та визнання пов'язаних з цим ризиків.

Цей етап дає чіткіше уявлення про масштаби всього проєкту та передбачувані питання, можливості та передумови виникнення проєкту.

Етап збору вимог необхідний для команди розробки, щоб отримати детальні та точні вимоги, яких слід дотримуватися у процесі розробки. Це допомагає компаніям доопрацювати оцінку та графік розробки визначеного додатку або системи.

Фаза 2: Техніко-економічне обґрунтування. Після завершення етапу аналізу вимог наступним кроком є визначення та документування вимог для визначеного програмного забезпечення. Цей процес проводиться за допомогою документа "Специфікація вимог до програмного забезпечення", також відомого як "SRS" (Software Requirement Specification). Він включає всі частини системи, що мають бути спроєктованими та розробленими протягом життєвого циклу проєкту.

Є п’ять показників для перевірки точності визначеного технікоекономічного обґрунтування:

1. Економічний: чи зможе команда завершити проєкт у межах закладеного бюджету?
2. Юридичний: чи зможе команда дотриматись вимог законів, що стосуються кібербезпеки та інших нормативно-правових актів або законів, що стосуються галузі у межах проєкту?
3. Функціональний: чи зможе команда створити функціонал, на який розраховує клієнт?
4. Технічний: перевірити, чи зможе поточна комп'ютерна система підтримувати програмне забезпечення?
5. Графік роботи: чи зможе команда реалізувати проєкт у межах визначеного і узгодженого терміну?

Фаза 3: Проєктування. На третьому етапі розробляється технічна документація та технічні специфікації програмного забезпечення відповідно до документа із вимогами. Це допомагає визначити загальну архітектуру системи.

Етап проєктування служить вхідним фактором для наступної фази моделювання. На цьому етапі розроблюється два види проєктних документів [11]:

1. Архітектура високого рівня (HLD – High-Level Design):
- короткий опис та назва кожного модуля;
- описи функціональності кожного модуля;
- співвідношення інтерфейсів та залежність між модулями;
- таблиці баз даних, визначені разом з їх ключовими елементами;
- повні діаграми архітектури разом з деталями технології.

2. Архітектура низького рівня (LLD – Low-Level Design):
- функціональна логіка модулів;
- таблиці баз даних, що включають тип та розмір;
- повна деталізація роботи інтерфейсу;

- перелік повідомлень про помилки;
- повний вхід і виходи для кожного модуля.

Фаза 4: Кодування. Після закінчення фази проєктування системи наступною фазою є кодування. На цій фазі розробники починають будувати всю систему, записуючи код з використанням обраної мови програмування. На етапі кодування завдання поділяються на блоки або модулі і призначаються різним розробникам. Це найдовша фаза процесу життєвого циклу розробки програмного забезпечення. На цій фазі розробнику потрібно дотримуватися певних заздалегідь визначених правил кодування, що забезпечить чітку і зрозумілу структуру і чистоту програмного коду, а також чітко дотримуватися технічних вимог, що в свою чергу впливає на продуктивність та відповідність кінцевого результатуочікуванням клієнта.

Фаза 5: Тестування. Після того, як написання програмного забезпечення буде завершено, його буде розгорнуто в тестовому середовищі. Команда тестування починає перевіряти функціональність всієї системи. Це необхіднодля того, щоб перевірити, чи працює вся програма відповідно до вимог замовника.

Під час цієї фази QA (Quality Assurance – спеціаліст, що відповідає за якість кінцевого продукту) та група тестувальників можуть виявити деякі помилки або дефекти, їх ще називають багами, про які вони повідомляють розробникам у вигляді звіту з детальним описом суті дефекту, кроків до його відтворення і середовища його виявлення. Команда розробників виправляє помилку та відправляє оновлений програмний код назад до QA для повторного тестування. Цей процес триває доти, доки програмне забезпечення не буде виправлено і відповідатиме критеріям якості, а версія продукту не буде стабілізована та працюватиме відповідно до бізнес-потреб системи.

Фаза 6: Встановлення / розгортання. Після завершення фази тестування програмного забезпечення, за умови, якщо в системі не залишиться явних критичних помилок або дефектів, розпочинається кінцевий процес розгортання.

На основі зворотного зв’язку, який надає менеджер проєкту, остаточна версія програмного забезпечення перевіряється на наявність проблем із розгортанням на стороні клієнта.

Фаза 7: Технічне обслуговування. Після розгортання клієнти починають використовувати розроблену систему в умовах максимально наближених до реальності. Цей момент дуже важливий у життєвому циклі програмного забезпечення, адже визначає наскільки система здатна працювати з реальними даними, навантаженнями і за реальних умов, а також чи вирішує вона поставлені бізнес-задачі. Для кращого налаштування кінцевого продукту під потреби клієнта виконуються наступні дії [11]:

1. Виправлення помилок – на етапі тестування спеціалісти могли не пропрацювати деякі специфічні сценарії, тому звіт про знайдені таким чиномбаги надсилатиме вже кінцевий користувач.
2. Оновлення – оновлення на наступну версію програмного забезпечення з виправленими помилками.
3. Покращення – додавання деяких нових функцій у існуюче програмне забезпечення або оптимізація роботи наявних.

Основна увага цієї фази SDLC приділяється забезпеченню того, щоб потреби не припиняли задовольнятися, і система продовжувала працювати відповідно до специфікацій, згаданих у першій фазі.Оскільки SDLC є лише концепцією, існує декілька популярних способів її реалізації через відповідні методи і методології розробки програмного забезпечення в аспекті управління проєктами. Нижче наведено основні методології, котрі використовуються для розробки програмного забезпечення в контексті SDLC.

1. Модель водопаду. Waterfall – широко відома модель SDLC. При такому підході весь процес розробки програмного забезпечення поділяється на різні фази. У цій моделі SDLC результат однієї фази виступає вхідним фактором для наступної фази. На рисунку 1.3 представлена схема процесу розробки за моделлю Waterfall.Ця модель SDLC є трудоємкою, адже на ранніх етапах детально документується все, що потрібно виконати на наступних

етапах. Вона була однією з перших описаних моделей, яка допомогла менеджерам проєктів структурувати роботу для їхніх команд. Ця модель найкраще пасує для урядових проєктів різного рівня, коли вимоги до кінцевого результату чітко визначені заздалегідь, детально задокументовані і не зазнають значних змін у процесі. Адже кожне потенційне доопрацювання, що відходить від першопочаткового плану завдає значних фінансових і часових втрат, адже модель абсолютно не орієнтована на зміну вимог.

Рисунок 1.3 – Схема процесу розробки за моделлю Waterfall [8]

3. Інкрементний підхід. Інкрементальна модель не є окремою моделлю – по суті це цикл водопадів. Вимоги поділяються на групи на початку проєкту. Для кожної групи використовується модель SDLC для розробки програмного забезпечення. Процес ітеративно повторюється, з кожним випуском додається більше функціональних можливостей, поки всі вимоги не будуть виконані. У цьому методі кожен цикл виступає фазою підтримки попереднього релізу програмного забезпечення. Модифікація інкрементної моделі дозволяє циклу розробки перекривати інші цикли. Внаслідок цього наступна ітерація може розпочатися до завершення попередньої. На рисунку 1.4 зображена схема процесу розробки за інкрементною моделлю.

Рисунок 1.4 – Схема процесу розробки за інкрементною моделлю [8]

3. V-модель. У цьому типі моделі SDLC етапи тестування і розробки плануються паралельно. Отже, є фаза верифікації з одного боку і фаза валідації з іншого. Розпаралелення процесу за V-моделлю починається після закінчення фази кодування. На рисунку 1.5 зображена схема життєвого циклу розробки програмного забезпечення за V-моделлю. Ця модель є еволюційним розвитком Waterfall. Хоча вона все ще погано орієнтована на зміну вимог, дозволяє запускати дві гілки процесів одночасно, що економить час на етапі тестування і виправлення помилок.Верифікація і валідація у цьому випадку доповнюють одна одну як дотримання логічних вимог і технічних аспектів. На схемі видно, що кожен із кроків гілки верифікації має відповідники у валідації, і навпаки. V-модель покликана тестувати продукт якомога раніше і ретельніше, що добре підходить для проєктів, для яких якість і безперебійність критичними вимогами, наприклад, медичні інформаційні системи.

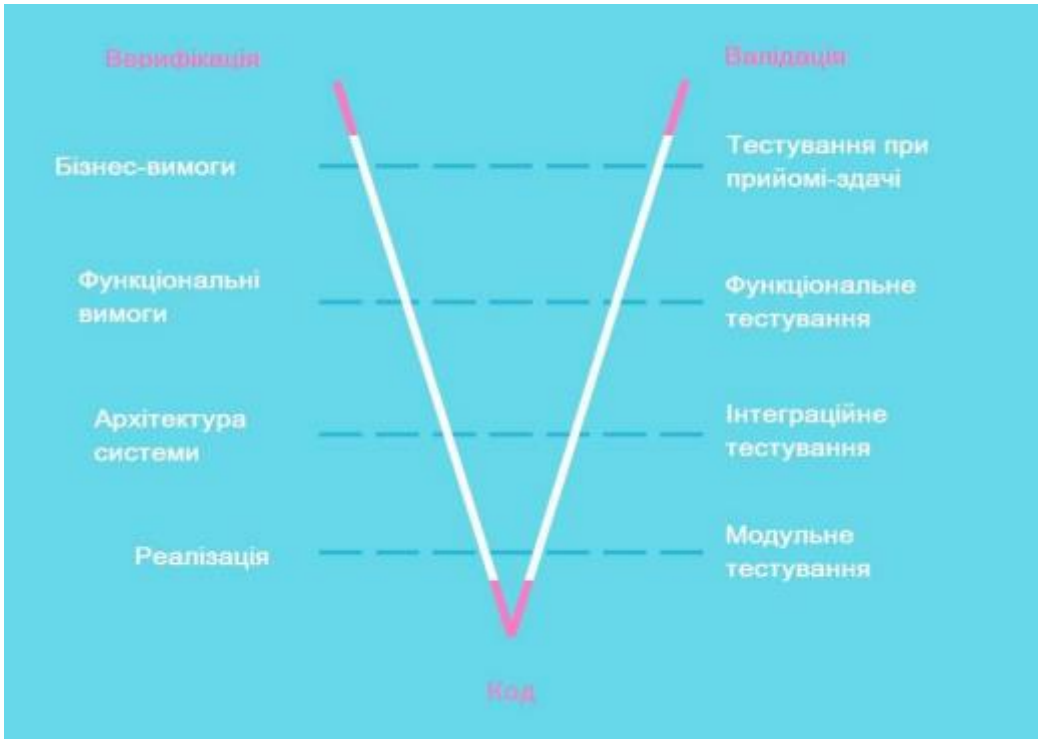


Рисунок 1.5 – Схема процесу розробки за V-моделлю [8]

4. Гнучка модель. Agile методологія – це практика, яка сприяє продовженню взаємодії розробки та тестування під час процесу SDLC будьякого проєкту. У методі Agile весь проєкт поділяється на невеликі часткові релізи. Усі ці складові розробляються в ітераціях, і кожна ітерація триває від одного до трьох тижнів. На рисунку 1.6 представлена загальна схема розробки за гнучкими методологіями. Гнучкі методології наразі найширше застосовуються для розробки програмних продуктів різних масштабів і призначення. Основна причина в тому, що вони зручні для замовника, адже легко підлаштовуються під зовнішні зміни. Вони орієнтовані на плідну співпрацю з клієнтом і зручність продукту для користувача. Група методологій Agile як великий фреймворк включає багато гнучких методологій, кожну зі своїми особливостями. Деякі з них буде детальніше розглянуто у межах цього підрозділу.

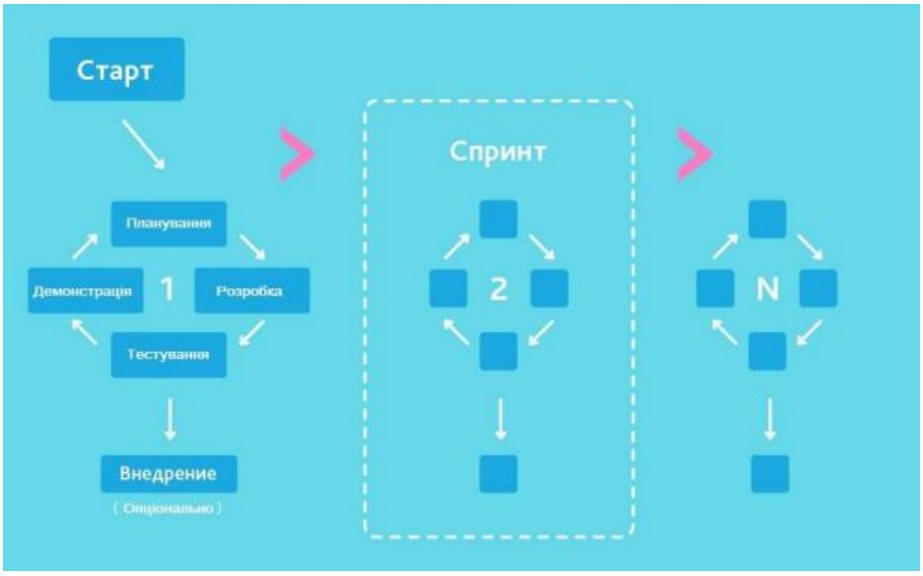


Рисунок 1.6 – Схема процесу розробки з використанням гнучких Методологій [8]

5. Спіральна модель. Спіральна модель – це модель, орієнтована на ризик. Ця модель SDLC допомагає команді прийняти елементи однієї або декількох моделей технологічних процесів, таких як водопад, інкрементна, гнучка тощо. Ця модель використовує найкращі характеристики моделі

прототипування та моделі водопаду. Спіральна методологія – це поєднання швидкого прототипування та одночасного виконання в проєктноконструкторській діяльності. Побудову такого процесу краще представити графічно, що зображено на рисунку 1.6.

6. Модель великого вибуху. Модель великого вибуху (Big Bang Model) орієнтована на всі типи ресурсів у розробці та кодуванні програмного забезпечення, взагалі без планування або з дуже малою його часткою. Вимоги оброблюються та реалізуються, коли вони надходять.

Рисунок 1.7 – Схема процесу розробки за спіральною моделлю [8]

Ця модель найкраще працює для невеликих проєктів із меншими за розмірами командами, які працюють разом. Вона також вдало підходить для академічних проєктів з розробки програмного забезпечення. Це ідеальна модель, коли вимоги або невідомі, або кінцева дата випуску не вказана. Вона дозволяє зосередити усі ресурси на етапі розробки і кодування, уникаючи при цьому значних ресурсних затрат на планування. Схема роботи цієї моделі зображена на рисунку 1.8. З наведеного вище можна зробити висновок, що вибір відповідної методології, яка буде застосовуватися на проєкті залежить винятково від цілей проєкту та очікуваного результату, а також виходячи із готовності зацікавлених сторін до регулярної взаємодії. З огляду на вибір методології управління проєктами значно залежить і успіх його виконання. Тому питання аналізу впливу методології проєктного менеджменту на результат розробки програмного забезпечення, особливо у малому та середньому бізнесі, є досить актуальним.

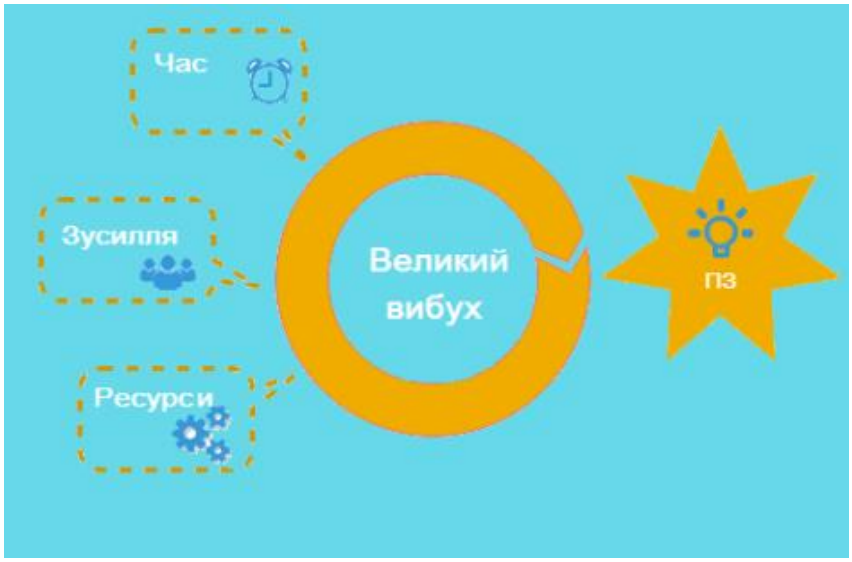


Рисунок 1.8 – Схема процесу розробки за моделлю великого вибуху [13]

Дослідження [14] з понад 10,5 тис. проєктів показало, що лише 2,5% компаній завершують свої проєкти абсолютно успішно. Решта або не виконали деякі свої першочергові цілі, або перевищили початковий бюджет чи терміни. Ці невдачі приносять великі втрати – лише в США невдалі IT-проєкти оцінюються сумарно 50 – 150 млрд. доларів втраченого доходу та продуктивності. А 17% IT-проєктів можуть піти настільки погано, що загрожують самому існуванню компанії.

Нині на ринку розробки IT-продуктів переважають «гнучкі» методології розробки. Найбільш використовуваним сімейством гнучких методологій є

Agile. Популярність цього фреймворку зумовлена мінливістю світу і різницею процесу розробки продукту в теорії і на практиці. У сучасному глобалізованому світі на успіх продукту, як і на його користувачів впливає безліч розрізнених факторів, тому власники продукту не можуть собі дозволити витрачати час і гроші на розробку нежиттєздатного продукту, доки ринок невпинно змінюється і вдосконалюється. Це добре простежується через 4 основні цінності Agile [15] :

1. Люди та співпраця важливіші за процеси та інструменти.

2. Працюючий продукт важливіший за вичерпну документацію.

3. Співпраця із замовником важливіша за обговорення умов контракту.

4. Готовність до змін важливіша за дотримання плану.

Scrum як методологія окреслює [16] :

1. Правила, згідно з якими перелік вимог до продукту повинен бути розроблений та керований у напрямку досягнення максимальної ефективності відповідно до функціональних можливостей.

2. Правила планування циклів розробки для забезпечення максимальної включеності команди задля отримання бажаного результату.

3. Основні правила взаємодії членів команди для оперативної реакції на непередбачувані робочі ситуації.

4. Правила аналізу і регулярного підлаштування процесу розробки для покращення аспектів взаємодії всередині команди. Кожну ітерацію можна охарактеризувати за послідовністю процесів: «Плануємо – Фіксуємо – Реалізуємо – Аналізуємо». Фіксація вимог в межах однієї ітерації та коригування тривалості ітерації дає змогу управляти балансом між гнучкістю та можливістю чітко спланувати розробку у короткостроковій перспективі. Методологія Scrum орієнтована на швидке пристосування до мінливих вимог, що дозволяє команді швидко підлаштовувати продукт під приватні потреби замовника. Слоган Scrum – «аналіз і адаптація»: проаналізувати отримане і адаптувати розроблене до реальних потреб, а потім все спочатку.

Що менше формалізму, то гнучкішою та ефективнішою буде співпраця, – базовий принцип цієї методології. Але це не означає, що формальності взагалі повинні бути виключені, вони мають застосовуватися в міру для створення сприятливого середовища ефективної взаємодії і проєктного менеджменту.

Інша популярна методологія у межах фреймворку Agile є Kanban. У 20% проєктів компанія використовує Kanban, при цьому 43% компаній стверджують, що використовують Kanban як одну із методологій на підприємстві [17].

В умовах системи Kanban, на відміну від традиційного підходу, команда не має завершеного плану й графіку роботи, а він, у свою чергу, тісно пов'язаний з окремим замовленням клієнта. Отже оптимізація роботи відбувається в межах замовлення. Графіки виробництва не створюються, а формуються виключно рухом карток Kanban. Розробка перебуває у постійному стані надбудови, системно адаптуючись до змін ринку. Можна сказати, що це процес без початку і кінця, адже робота на конкретний період визначається лише поточними потребами і пріоритетністю задач, що ці потреби покривають.

Методологія розробки програмного забезпечення Kanban була застосована у практичній діяльності Девідом Андерсеном у 2007 році. Багато з цих методів та підходів використовувались різними командами Agile, перш ніж були описані загалом. Новизною стало запровадження типу завдань "в процесі виконання". Інші команди Agile робили це і раніше, але в Kanban існує відомий ліміт на кількість завдань, які можна виконувати за одиницю часу. Цей ліміт зазвичай досить низький – обмеження за кількістю приблизно таке ж, як кількість програмістів у команді, іноді навіть менше.

Основні положення Kanban [16]:

1. Візуалізація процесу робіт:

а) розбиття роботи на частини, коли кожна частина вписується у картку і прикріплюється до дошки;

б) розбиття задач на колонки для розділення на стадії розробки.

2. Встановлення обмеження на незавершені задачі (work-in-progress), коли визначається можлива кількість не доведених до кінця робіт на кожному етапі робочого процесу.

3. Вимірювання часу, затраченого на виконання завдання (lead time) (середня тривалість часу для завершення одного пункту, іноді звана "оперативним часом"), вдосконалення процесу, для мінімізації часу виконання завдання і максимально можливого прогнозування його виконання за

ресурсами. Kanban – це не окремо взятий конкретний процес, а узагальнена система цінностей.

Нині на підприємствах для полегшення побудови процесу за гнучкими методологіями використовуються так звані таск- та баг-трекінгові системи, за допомогою яких спрощено можливість підрахунку показників діяльності проєкту та ведення звітності, а також відстеження ефективності команди та руху дорожньою картою проєкту відносно поставлених цілей та визначених задач.

Однак перш ніж з’явилися гнучкі методології проєктного менеджменту, попередньою ітерацією розвитку були каскадні методології. Класична каскадна методологія носить назву Waterfall – «Водопад», від якої беруть початок усі інші каскадні, а згодом усі інші методології.

Етапи оригінального методу «Водопаду», розробленого Ройсом, включають визначення вимог, проєктування, впровадження, перевірку та обслуговування. Інші моделі змінюють фазу вимог на фазу ідеї або розбивають етап вимог на планування та аналіз. Крім того, деякі моделі далі розбивають етап проєктування на підфази логічного та фізичного проєктування. Однак, основні принципи залишаються тими ж.

Метод «Водопаду» передбачає, що всі вимоги можуть бути зібрані напередодні першого етапу. Спілкування з користувачем передує на цій фазі, оскільки менеджер проєкту робить все можливе, щоб отримати детальне розуміння вимог користувача. Після завершення цього етапу процес проходить угору.

Фаза проєктування найкраще описана шляхом її розбиття на підфази логічного дизайну та фізичного дизайну. Під час фази логічного проєктування аналітики системи використовують інформацію, зібрану на етапі вимог, для розробки системи незалежно від будь-якої апаратної чи програмної складової. Після завершення логічного дизайну вищого рівня системний аналітик починає перетворювати його на фізичний дизайн, що залежить від специфікацій конкретних апаратних та програмних технологій.

Фаза реалізації – це етап, на якому пишеться весь фактичний код. Цей етап розробки належить програмістам, оскільки вони приймають вимоги та технічні характеристики проєкту та пишуть код програми.

Етап верифікації спочатку був ініційований Ройсом, щоб переконатися, що проєкт відповідає очікуванням клієнтів. Однак під час реального аналізу та проєктування системи цим етапом часто нехтують. Проєкт розгортається для замовника на реальному середовищі, і починається етап технічного обслуговування. Це одна із причин, чому гнучкі методології витісняють каскадні на ринку IT-послуг.

Під час фази технічного обслуговування замовник використовує розроблену систему або додаток. Оскільки проблеми з'являються через некоректне визначення вимог, через помилки в процесі проєктування або через зміну вимог кінцевих користувачів, протягом цієї фази в систему вносяться зміни.

Метод «Водопаду» має певні переваги, зокрема[18]:

- Помилки проєктування фіксуються до написання будь-якого програмного забезпечення, заощаджуючи час на етапі впровадження.
- Детально складена технічна документація є одним із результатів підходу, і новим програмістам легше пристосуватися на етапі технічного обслуговування.
- Підхід напрочуд структурований, і простіше виміряти прогрес, посилаючись на чітко визначені віхи.
- Загальну вартість проєкту можна точно оцінити після того, як будуть визначені вимоги (за допомогою функціональних та технічних специфікацій користувальницького інтерфейсу).
- Тестування простіше, оскільки це можна зробити, посилаючись на сценарії, визначені у функціональній специфікації.

На жаль, метод «Водопаду» має в собі чимало недоліків, таких як [18]:

- Клієнтам часто буде складно викладати свої побажання на абстрактному рівні функціональних вимог, і вони можуть оцінити результат у

повній мірі лише після наочного користування програмою. Зміни функціоналу на кінцевих етапах розробки додатку є трудоємкими і дорогими.

- Модель не забезпечує можливість зміни вимог протягом циклу розробки.
- Часто проєкт може зайняти значно більше часу, ніж коли він розробляється за допомогою ітеративної методології.

Через ряд подібних проблем системні аналітики почали шукати альтернативні методи проєктування систем, що поклало початок розвитку гнучких ітеративних методологій, які вирішують багато проблем каскадної розробки.

Якщо розглядати дві великі групи методологій проєктного менеджменту – каскадні і гнучкі – у розрізі трьох основних точок проєктного менеджменту, так званого трикутника, то побачимо переваги і недоліки обох умовних груп методологій, Таблиця 1.3 :

1. Час. З дотриманням чітких вимог на перших етапах каскадна модель витрачає набагато менше часу, щоб забезпечити необхідний хід робіт, аніж того вимагають перші кроки з використанням гнучких методологій.

Однак, рухаючись вглиб проєкту стає зрозуміло, що кожне відхилення від початкових вимог займе відповідно багато часу на переробку при каскадній моделі.

2. Бюджет. При неможливості повернення до попереднього етапу у каскадній моделі, кожна поправка і відхилення коштуватиме проєкту значну суму, не передбачену у бюджеті, на відміну від гнучких методологій, які орієнтовані на мінливість ринку і вимог.

3. Зміст. Оскільки гнучкі методології потребують постійної роботи у стані невизначеності, вони потребують також і значного організаційного впливу на команду розробки, а також врахування ризиків при плануванні.

Однак мета гнучких методологій – показати готові модулі системи після кожної ітерації робіт. Відповідно, вносити зміни буде набагато дешевше і швидше, ніж у каскадних моделях.

Таблиця 1.3 Порівняння найпопулярніших методологій проєктного менеджменту [19]

Критерії	*Методології*		
	Scrum	*Kanban*	*Waterfall*
Час і бюджет			
Визначений заздалегідь бюджет			+
Визначені заздалегідь вимоги			+
Включає оцінку завдань	+		+
Процес			
Постійна взаємодія з замовником	+	+	
Має чітко визначені ролі	+		+
Ітеративний підхід	+	+	
Обмеження на кількість завдань в одиницю часу		+	
Завдання можуть виконуватись одночасно	+	+	
Продукт			
Якісний продукт у результаті	+	+	+
Демонструє проміжні результати	+		
Включає постійне покращення	+	+	

Висновки до розділу 1

Наразі існує чимало відомих методологій та фреймворків проєктного менеджменту. Декотрі втратили свою актуальність, натомість донині точиться битва між вибором гнучких та каскадних методологій. Для різних проєктних цілей необхідно застосовувати різні інструменти досянення мети. Під визначення інструментів у даному сенсі підпадають методології проєктного менеджменту, які зазвичай можуть неточно підходити під окремі IT-проєкти за сукупністю таких причин:

- виникнення провторюваних управлінських часових і ресурсних затрат через переформатування або повторну перевірку цілей методології;
- виникнення провторюваних управлінських часових і ресурсних затрат через зміну порядку проєктних робіт або зміни у ресурсній базі;
- бажання замовника регулярно контролювати команду розробки і висловлювати власну думку відносно отриманого результату;
- особливості взаємодії спеціалістів всередині команди та людський фактор;
- зміни на ринку IT-послуг, тенденцій та трендів, нові релізи продуктів конкурентів;
- наявність поправок до проєкту, отриманих після старту розробки, що змінють дорожну мапу і ресурсну навантаженість проєкту.

Таким чином, перед нами постає завдання автоматизувати повторювані ітерації проєктного менеджера з урахуванням оптимізації вибору методології, проєктних робіт, запобігання помилок та простоїв команди, і як наслідок – економія бюджету і швидкість забезпечення робочого кінцевого продукту, що відповідає першопочатковим цілям і задовольняє потреби замовника та цільової аудиторії.

РОЗДІЛ 2 ІНФОРМАЦІЙНИЙ АНАЛІЗ ПРОЄКТНОЇ ДІЯЛЬНОСТІ ПІДПРИЄМСТВА

2.1 Вирішення завдань проєктного менеджменту шляхом застосування нейронних мереж

Все частіше у царині проєктного менеджменту нейронні мережі широко використовуються для таких складних задач, як передбачення ризиків,складання бюджету або вибір потенційно успішного проєкту з портфелю для інвестування. Проаналізуємо ці випадки. Першим кроком для розробки нейронної мережі є підготовка базового набору даних, який буде використовуватися як орієнтир для тренування нейронної мережі. Важливо підкреслити, що зазвичай правильний набір даних є дорогим і вимагає багато часу для збору. Набір даних складається з набору змінних, заповнених інформацією, яка буде використовуватися в якості посилання. У проєктному середовищі для обчислення бюджету проєкту можна використовувати кілька змінних. Деякі поширені приклади [12]:

1. Складність: рівень складності проєкту (низький, середній, високий). Зазвичай це незалежна категорія.
2. Бюджет: запланований бюджет проєкту. Це числова змінна, яка може бути незалежною або залежною (вихід).
3. Фактична вартість: фактичні витрати проєкту. Здебільшого це
4. незалежна числова змінна.
5. Варіант витрат: різниця між бюджетом і фактичною вартістю. Це числова змінна, яка може бути незалежною або залежною (вихід).
6. Базова тривалість: оціночна або середня тривалість типового проєкту. Це незалежна числова змінна.
7. Фактична тривалість: фактична тривалість проєкту. Зазвичай це незалежна числова змінна.
8. Відхилення тривалості: різниця між базовою тривалістю та фактичною тривалістю.

9. Тип договору: незалежна змінна категорія, яка визначає тип договору, що використовується для робіт у проєкті (наприклад, фіксована ціна, вартість плюс, ціна одиниці).

10. Кількість відповідних груп зацікавлених сторін: незалежна числова змінна, що відображає кількість відповідних груп зацікавлених сторін у проєкті. Коли набір даних готовий, мережа готова пройти навчання. Для навчального процесу можуть використовуватися два підходи: навчання під керівництвом або адаптивне навчання.

Однією з найбільших проблем методу навчання є визначення, яку мережу використовувати та процес її технічного налаштування. Деякі мережі можуть бути навчені за лічені секунди, але в деяких складних випадках з багатьма змінними та виходами можуть знадобитися години для виконання навчального процесу.

Результати навчального процесу – це складні формули, які пов'язують вхідні чи незалежні змінні з виходами (надійні змінні), як графік. Більшість комерційних програмних пакетів зазвичай перевіряють результати навчання за допомогою деяких точкових даних, щоб оцінити якість навчання. Близько 10–20% датасету використовується для тестування.

Після тренування модель готова передбачити майбутні результати.

Найбільш релевантною інформацією, яка повинна бути зосередженою у дослідженні, є внесок кожної окремої змінної у передбачувані результати та надійність моделі. Використання штучних нейронних мереж може бути корисним інструментом для визначення показників бюджету проєкту, таких як вартість управління проєктом, орієнтовна вартість пропозиції постачальника або вартість страхування обладнання. Нейронні мережі забезпечують точний процес прийняття рішень без алгоритму чи процесу на основі формули.

З недавньою розробкою програмних засобів процес обчислення стає дуже простим та зрозумілим. Однак найбільша проблема у створенні достовірних результатів полягає в якості відомої інформації. Весь процес

заснований на фактичних результатах, і зазвичай найдорожча та кропітка частина процесу пов'язана з отриманням достовірних даних для навчання та тестування процесу.

Однією з головних турбот проєктних менеджерів є те, що вони достеменно не знають, чи буде проєкт під їх керівництвом успішним чи ні.

Однак це може базуватися не лише на здогадці. Ці показники цілком реально відстежувати інструментами контролю та моніторингу, визначеними в структурах управління проєктами та органами знань. Як було зазначено у першому розділі, проєкт, традиційно, може вважатися успішним, якщо він досягнув поставленої мети у межах трикутника проєктного менеджменту: обсяг, бюджет та графік. Для цього було б достатньо традиційної статистики або параметричних інструментів. Однак ці інструменти залишають осторонь інші якісні аспекти управління проєктами, наприклад, точку зору зацікавлених сторін.

Вибір проєктів є проблемою тактичного рішення з характеристиками декількох, суперечливих та непропорційних критеріїв, тоді як ті, хто приймає рішення, приймають рішення щодо портфоліо найбільш привабливих варіантів, розглядаючи різні сторони щодо ефективності проєктів.

Відбір проєктів вважається складною процедурою прийняття рішень, оскільки на нього впливають суттєві рішучі фактори, включаючи умови ринку, доступність сировини та ресурсів, шанс практичного успіху та політику уряду З наявністю обмежених ресурсів для вибору та управління проєктами, керівники команд стикаються з різними труднощами, пов'язаними з аналізом, ранжуванням та відбором проєктів. Зрозуміло, що помилкові рішення щодо вибору проєктів мають два суворих наслідки: по-перше, ресурси вичерпуються неналежними проєктами, по-друге, організація втрачає прибуток, який вона могла б отримати, якби ресурси були витрачені на інших успішних проєктах.

Нейронні мережі найкраще справляються із поставленими задачами в умовах великої кількості вхідних показників і слабкого зв’язку між змінними.

Так серед вхідних даних можуть бути такі, які на перший погляд здаються незначними для результуючого показника, однак на практиці у поєднанні з іншою змінною дають високу точність у прогнозуванні вихідної змінної. В контексті методології проєктного менеджменту нейронна мережа дозволить підвищити точність вибору необхідної методології, що в свою чергу звільняє час менеджерів від рутинного аналізу вхідних даних проєкта для вибора відповідної моделі взаємодії. Все, що вимагається від менеджера – подати на вхід попередню оцінку проєктної команди стосовно проєкту. Це дозволить не лише оптимізувати час роботи проєктного менеджера, але й запобігти потенційним втратам у часі і бюджеті, адже неправильно обрана методологія може суттєво знизити продуктивність команди і зіпсувати стосунки із клієнтом.

Для поставленої у цій роботі мети необхідно використовувати вхідний сет даних, а отже мається на увазі навчання з учителем. Вибір методології проєктного менеджменту має на увазі передбачення категоріальної змінної, тому йдеться про задачу класифікації. У якості алгоритму, як було описано вище, розглянемо нейронну мережу.

2.2 Аналіз моделі побудови нейронної мережі для аналізу проєкту

Нейронна мережа (Artificial Neural Network) має здатність вивчати закономірності на прикладах. ANN – модель обробки інформації, натхненна біологічною системою нейронів. Вона складається з великої кількості сильно взаємопов'язаних елементів обробки, нейронів, для вирішення задач. Модель слідує нелінійному шляху і обробляє інформацію паралельно по всіх вузлах [8].

Нейронна мережа – це складна адаптивна система. Адаптивна означає, що вона має здатність змінювати свою внутрішню структуру, регулюючи ваги входів.

Нейронна мережа була розроблена для вирішення проблем, які легкі для людей і важкі для машин, наприклад, ідентифікація фотографій кішок і собак,

ідентифікація пронумерованих зображень. Ці проблеми часто називають розпізнаванням образів. Його застосування варіюється від оптичного розпізнавання символів до виявлення об'єктів.

Нейронні мережі виконують обчислення за допомогою процесу навчання.

Нейронна мережа являє собою набір підключених входів / виходів, в яких кожне з'єднання має вагу, пов'язану з ним. На етапі навчання мережа оброблює дані, регулюючи вагові коефіцієнти для прогнозування правильної мітки класу вхідних даних.

Мозок людини складається з мільярдів нейронних клітин, які обробляють інформацію. Кожна нейронна клітина розглядається як проста система обробки.

Біологічна нейронна мережа, передає інформацію через електричні сигнали. Ця паралельна інтерактивна система змушує мозок мислити і обробляти інформацію. Дендрити нейрону отримують вхідні сигнали від іншого нейрона і реагують на вихід на основі цих входів до аксона деякого іншого нейрона.

Виходячи з цих входів, відбувається передача вихідного сигналу через аксон.

1) Збір даних

Спочатку необхідно визначити, які дані будуть представляти об’єкт в майбутній моделі. По-перше, потрібно не пропустити значущі описові характеристики об’єкта, по-друге, встановити фіксовані критерії для прийняття рішення про зазначену ознаку. Виділяють три основні категорії змінних:

a) булеві (бікатегоріальні), значенням яких є «true» або «false», 1 або 0.
b) категоріальні, значенням яких є слово або словосполучення, що представляє певний клас. Зазвичай класів буває більше двох (мультикатегоріальні), у іншому випадку їх можна звести до

булевої категорії змінних. Наприклад, форма: трикутник, квадрат або коло.

c) кількісні, значенням яких є цифри і числа, що визначає певну шкалу. Наприклад, сума годин, витрачених на розробку проєкту.

При побудові моделі слід враховувати, з якими типами змінних алгоритм може мати справу. Наприклад, дерево рішеннь здатне навчатися на будь-яких типах змінних, а от нейронна мережа приймає тільки числову категорію вхідних даних і навчається лише із застосуванням кількісних ознак. Тобто для опрацювання нейронною мережею необхідно замінити категоріальні змінні на числові.

2) Препроцесинг.

Після збору даних, необхідна їх первинна обробка, щоб привести до прийнятного алгоритмом вигляду. Основна мета препроцесингу – відобразити дані у форматі придатному для навчання конкретної моделі. Можна виділити три основних маніпуляції над даними на етапі препроцесингу:

А) Створення векторів ознак для прикладів тренувальної вибірки.

Загальними словами, це процес має на меті привести вхідні дані в числову форму. Таким чином можна позбавитися від булевих, категоріальних і інших нечислових типів.

Б) Нормалізація даних. Процес, при ми штучно змінюємо статистичні характеристики вибірки, щоб середнє арифметичне дорівнювало нулю, а дисперсія – одиниці. Є різні методи нормалізації даних, але усі вони переслідують однакову мету – зробити вхідні дані придатними для обробки алгоритмом.

В) Зміна розмірності векторного простору. Якщо векторний простір ознак занадто великий (мільйони ознак) або малий (менше десятка), застосовуються методи підвищення або зниження розмірності простору.

Для підвищення розмірності можна використовувати частини навчальної вибірки як опорні точки, додавши в вектор ознак відстань до цих

точок. Цей метод часто призводить до того, що в просторах вищої розмірності множини лінійно розділяються, що спрощує завдання класифікації.

Для зниження розмірності найчастіше використовують PCA. Основне завдання методу головних компонент – пошук нових лінійних комбінацій ознак, уздовж яких максимізується дисперсія значень проекцій елементів навчальної вибірки.

Існує велика кількість алгоритмів машинного навчання, на основі яких можна побудувати моделі, наприклад:

- Decision Tree (дерево прийняття рішень),
- Random Forest (складається з окремих Decision Tree)
- KNN (метод k-найближчих сусідів),
- SVM (метод опорних векторів),
- NN (нейромережа).

Вибір моделі залежить від очікуваного результату. Необхідно визначити, наскільки важлива можливість інтерпретувати структуру моделі.

Обираємо Random Forest як алгоритм машинного навчання для проєкту.

Алгоритм Random Forest можна описати наступним чином:

1. Вибір кількості дерев : Визначаємо, скільки дерев будемо використовувати в лісі.

2. Для кожного дерева b=1,2,...,Bb = 1, 2, ..., Bb=1,2,...,B:

a. бутстреп-вибірка: Вибираємо випадкову вибірку даних з початкового набору даних;

b. побудова дерева починаючи з кореневого вузла, на кожному вузлі;

Випадково вибираємо підмножину ознак розміру m. З цієї підмножини ознак вибираємо ту, яка найкраще розщеплює дані згідно з обраним критерієм (наприклад, індексом Джині).

Далі розщеплюємо вузол на два дочірніх вузли. Продовжуємо цей процес, поки не буде досягнуто максимальна глибина або іншої умова зупинки.

3. Прогнозування для нового зразка x:

a. пропускаємо x через кожне дерево $h_b(x)$ і отримуємо прогноз;

b. класифікація: Вибираємо клас, який отримав найбільшу кількість голосів;

Схематично архітектура нейронної мережі представлена на рис. 2.1.

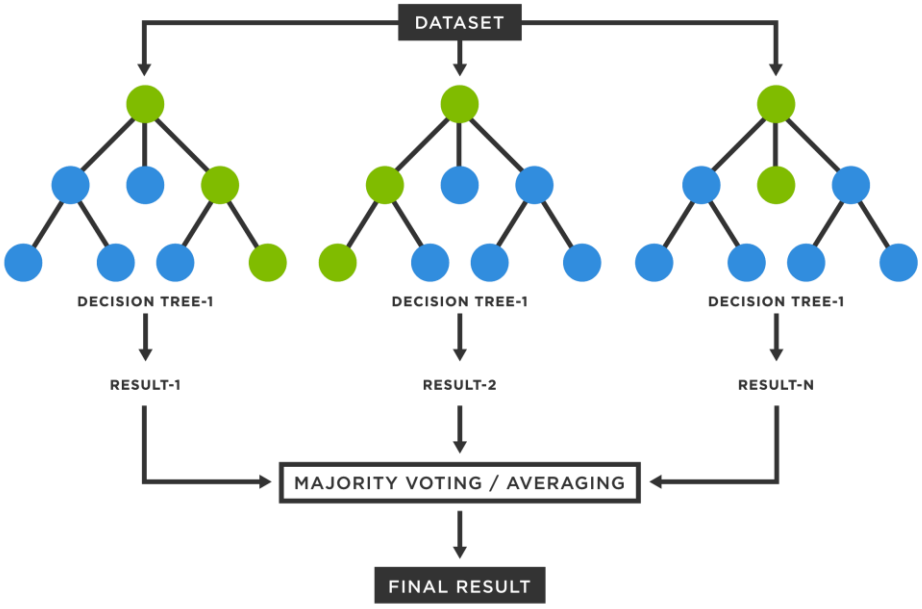


Рисунок 2.1 – Схема архітектури нейронної мережі [20]

2.3 Аналіз датасету вхідних даних проєктів для нейронної мережі

Проєктна діяльність кожної компанії зберігається у секреті і є внутрішнім надбанням компанії, що містить в собі профільну інформацію по основним бізнес-процесам компанії, тому визначена як конфіденційна інформація, і відповідно не підлягає розголошенню. Сет даних був зібраний із приватних джерел.

Кожна компанія провадить проєктну діяльність відповідно до внутрішніх процесів та процедур. Звісно, існують загальноприйняті найкращі практики, які використовуються прогресивними організаціями як дороговкази і рекомендації на шляху розробки продуктів та надання послуг. Однак,

єдиного глобального дослідження і відповідно набору даних з цієї теми у відкритих джерелах немає.

Дані для дослідження було отримано під час виробничої практики на підприємстві ТОВ «Прогрестех-Україна». Компанія працює на ринку IT та інженірингових послуг.

Датасет котрий був підготований для дослідження включає в себе Таблиця 2.1:

1. Розмір команди;
2. Бюджет;
3. Тривалість;
4. Зміна початкових вимог;
5. Складність виконаного проєкту;
6. Ризики;
7. Тип проєкту;
8. Методологія;

Як було зазначено у попередньому розділі роботи, найпоширенішими методологіями нині є Scrum, Kanban і Waterfall. Для препроцесингу даних використовувався скрипт на мові програмування Python. Для обробки датасету використовувалась багатофункціональна бібліотека для аналітики даних Pandas.

Однією з найпопулярніших багатофункціональних бібліотек для аналітики і обробки даних є Pandas – швидкий, потужний, гнучкий, простий у використанні інструмент з відкритим кодом для аналізу та маніпуляції, побудований на основі мови програмування Python. Pandas дозволяє виконувати наступне [2]:

Таблиця 2.1 Набір даних виконаних проєктів

Набір даних виконаних проєктів								
Шифр проєкту	*Розмір команди чол.*	*Бюджет тис. у.о.*	*Трив., міс.*	*Зміна вимог*	*Складн.*	*Ризик*	*Тип проєкту*	*Метод.*
Проєкт 1	6	500	10	Так	Висока	Високі	Новий Дизайн	Scrum
Проєкт 2	20	120	3	Ні	Низька	Низькі	Підтримка виробництва	Waterfall
Проєкт 3	14	200	12	Так	Середня	Середні	Технічна підтримка	Kanban
Проєкт 4	8	250	9	Так	Висока	Високі	Новий Дизайн	Scrum
Проєкт 5	18	100	2	Ні	Низька	Низькі	Підтримка виробництва	Waterfall
Проєкт 6	12	190	15	Так	Середня	Середні	Технічна підтримка	Kanban

Висновки до розділу 2

Проєктні менеджери з самого початку існування цієї професії стикаються з величезною кількістю факторів невизначеності, які зусібіч впливають на результат проєкту. Учені не менше 20 років шукають способи оптимізації роботи проєктних менеджерів шляхом застосування методик машинного навчання.

Найліпше для цих цілей підходить нейронна мережа, адже вона може приймати багато невзаємопов’язаних факторів на вхід, знаходячи між ними приховані закономірності. Для створення нейронних мереж існує багато відповідних інструментів, зокрема відкриті мови програмування і бібліотеки.

Проблема вибору методології проєктного менеджменту підпадає під задачу класифікації. Для будь-якої нейронної мережі необхідно підготувати вхідні дані, аби отримати найточніший результат.

РОЗДІЛ 3 ПОБУДОВА МОДЕЛІ ДЛЯ АВТОМАТИЗАЦІЇ ВПРОВАДЖЕННЯ МЕТОДОЛОГІЇ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ

3.1 Математичні основи побудови моделі автоматизації впровадження методології

Отже розглянемо математичну модель обраного в попередніх розділах алгоритму Random Forest :

1. Алгоритм Random Forest використовує метод бутстреп-агрегації для створення множини різних навчальних наборів даних:

З початкового набору даних $D = \{(x_i, y_i)\}_{i=1}^{N}$ випадково з поверненням вибирається N зразків, щоб створити навчальну вибірку для кожного дерева. Це означає, що деякі зразки можуть повторюватися, а деякі можуть не потрапити до вибірки.

2. Побудова дерев рішень:

При кожному розщепленні вузла дерева вибирається випадкова підмножина ознак mmm (де m<M, M — загальна кількість ознак). Це зменшує кореляцію між деревами та підвищує різноманітність моделі.

Дерево будується до максимальної глибини без обрізання, що дозволяє кожному дереву бути повністю розгалуженим.

3. Прогнозування:

Для класифікації: Кожне дерево дає свій прогноз класу $h_b(x)$для вхідного зразка x. Остаточний прогноз визначається шляхом більшості голосів серед усіх дерев:

$$y = mode\{h_b(x)\}_{b=1}^{B} \quad (3.1)$$

де B — кількість дерев у лісі.

Для регресії: Прогнозом є середнє значення прогнозів усіх дерев:

$$y = \frac{1}{B}\sum_{b=1}^{b} h_b\,(x) \quad (3.2)$$

4. Критерії розщеплення:

Для класифікації: Використовуються критерії, такі як індекс Джині (3.3) для визначення найкращого розщеплення вузла.

$$G(t) = 1 - \sum_{k=1}^{K} p(k|t)^2 \qquad (3.3)$$

де p(k|t)— частка зразків класу k у вузлі t, K — кількість класів.

Для регресії: Застосовується середньоквадратична помилка або інші метрики для оцінки якості розщеплення.

5. Важливість ознак

Random Forest дозволяє оцінити важливість ознак на основі того, наскільки вони зменшують критерій розщеплення в кожному дереві. Сумарне зниження критерію по всіх деревах дає міру важливості кожної ознаки.

6. Математичне обґрунтування

Random Forest можна розглядати як окремі Decision Trees. Об'єднуючи їх прогнози, ми отримуємо модель з низьким зсувом і низькою дисперсією завдяки усередненню.

Структура даної моделі включатиме 7 вхідні змінні та одну вихідну. Як було зазначено під час аналізу датасету, значення методології, яке необхідно передбачити має небінарний характер, оскільки ми розглядаємо три можливі категорії методології. Саме тому колонку виводу необхідно представити у вигляді вектора. Таким чином ми отримуємо 7 вхідні змінні та 3 вихідних.

Нейронна мережа повинна виявити приховані зв’язки між змінними і розрахувати дані виходів. Процес знаходження виходу в нейронної мережі називається процесом прямого поширення.

Оскільки Random Forest обрано як алгоритм машинного навчання в попередньому розділі і розглянуто його математичну модель в даному розділі – перейдемо до реалізації алгоритму за допомогою обраної для реалізації даного проєкта мови програмування Python.

В першій частині програми виконуємо імпортування бібліотеки Pandas і присвоюємо їй псевдонім pd.

import pandas as pd

Далі за допомогою функції train_test_split розділяємо дані на тестову і навчальну частину відповідно.

from sklearn.model_selection import train_test_split

Після цього обираємо клас RandomForestClassifier для створення моделі на основі алгоритму Random Forest.

from sklearn.ensemble import RandomForestClassifier

Імпортуємо функцію accuracy_score для обчислення точності класифікації.

from sklearn.metrics import accuracy_score

Створюємо набір даних на основі обраних показників.

data = { ... }

Створюємо таблицю, що буде містити дані з попереднього датасета

df = pd.DataFrame(data)

Перетворюємо текстові дані в числові, оскільки алгоритми машинного навчання більш коректно працюють з числовими даними.

df['Методологія'] = df['Методологія'].map({'Scrum': 0, 'Waterfall': 1, 'Kanban': 2})

Видаляємо змінну 'Методологія' з таблиці, оскільки цей параметр ми будемо передбачати

X = df.drop('Методологія', axis=1)

Створюємо змінну 'Методологія', котра буде прогнозуватися.

y = df['Методологія']

Дана функція ділить дані на тренувальний набір (80%) і тестовий набір (20%)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Ініціалізуємо модель RandomForestClassifier, обмежуємо глибину кожного дерева до 5 рівнів, щоб запобігти перенавчанню. А кожне дерево ділиться на нові вузли, якщо у вузлі є щонайменше 2 зразки.

tree_model = RandomForestClassifier(max_depth=5, min_samples_split=2, random_state=42)

Модель тренується на тренувальних даних X_train та y_train

tree_model.fit(X_train, y_train)

Після тренування модель використовується для прогнозування цільової змінної (методології) для тестового набору даних X_test.

y_pred_tree = tree_model.predict(X_test)

Оцінюється точність моделі, порівнюючи передбачені значення y_pred_tree з фактичними значеннями y_test. Функція accuracy_score повертає частку правильно передбачених результатів.

accuracy_tree = accuracy_score(y_test, y_pred_tree)

Виводиться точність моделі у відсотках з двома знаками після коми.

*print(f'Точність моделі Random Forest: {accuracy_tree * 100:.2f}%')*

В цій частині коду вводимо новий набір даних для прогнозування методології для нього, що представляє один новий проєкт.

новий_проєкт = pd.DataFrame([[ ]], columns=X.columns)

Використовується навчена модель для прогнозування методології для нового проєкту.

прогноз_tree = tree_model.predict(новий_проєкт)

Виводиться рекомендована методологія для нового проєкту на основі передбачення моделі (0 — Scrum, 1 — Waterfall, 2 — Kanban).

print(f'Рекомендована методологія для нового проєкту: {прогноз_tree[0]}')

3.2 Постановка гіпотези щодо впливу показників на вибір методології проєктного менеджменту у IT-проєктах

На початку тренування моделі часто важко передбачити, які саме показники будуть визначними у прогнозуванні результату. Виходячи із даних у датасеті та їх аналізу було висловлене припущення, що на вибір методології проєктного менеджменту впливають такі показники:

- Розмір команди
- Бюджет
- Тривалість проєкту
- Можлива зміна вимог
- Орієнтовна складність
- Можливі ризики
- Тип проєкту

Доповнимо частину коду представленого в попередньому розділі, що відповідає за датасет набором даних з Таблиці 2.1.

```
data = {
    'Розмір_команди': [6, 20, 14, 8, 12], # Спеціалістів
    'Бюджет': [500000, 120000, 200000, 250000, 190000], # Умов. Од.
    'Тривалість': [10, 3, 12, 9, 15], # Місяців
    'Зміна_вимог': [1, 0, 1, 1, 1], # 0- Ні, 1- Так
```

'Складність': [3, 1, 2, 3, 2], # 1- Низька, 2- Середня, 3- Висока

'Ризики': [3, 2, 3, 1, 2], # 1- Низькі, 2- Середні, 3- Високі

'Тип_проєкту': [1, 2, 3, 1, 3], # 1- Новий Дизайн, 2- Підтримка виробництва, 3- Технічна підтримка

'Методологія': ['Scrum', 'Waterfall', 'Kanban', 'Scrum', 'Kanban']

}

Доповнимо частину коду представленого в попередньому розділі, що відповідає за датасет набором даних з Таблиці 3.1.

новий_проєкт = pd.DataFrame([[18, 100000, 2, 0, 1, 1, 2]], columns=X.columns)

Таблиця 3.1 Набір даних нового проєкту

Набір даних нового проєкту							
Назва	*Розмір команди чол.*	*Бюдж. тис. у.о.*	*Трив., міс.*	*Зміна вимог*	*Складн.*	*Ризики*	*Тип проєкту*
Новий проєкт	18	250	18	Ні	Низька	Низькі	Підтримка виробн.

Проаналізувавши показники закономірним з логічної точки зору здається те, що проєкти, що підпадають під класифікацію Waterfall є дуже короткими за термінами – адже що коротший такий проєкт, то менший ризик отримати кінцевий продукт, відмінний від бажаного. Адже вимоги впродовж виконання проєкту не повинні змінюватися, отже вони мають бути чітко сформульовані і вкрай точно визначені на початковому етапі. Що стосується Kanban, теж цілком логічним здається те, що ця методологія поступилася Scrum, адже Kanban більш орієнтований на безперервний процес, ніж на чіткий результат, тому краще пасує для проєктів підтримки – які, очевидно, займають менше часу, ніж проєкти розробки. І лідер за усіма усередненими показниками – Scrum –методологія, що найкраще підходить для комунікації

як у команді, так і з замовником і полегшує розробку продукту, до якого можна доторкнутися і перевірити вже після перших декількох ітерацій.

Отже, мета перевірки даних у нейронній мережі полягає у знаходженні закономірностей у таких, здавалося б простих метриках проєкту, які допоможуть уникнути перевищених витрат або протермінування проєкту, а отже наблизять його шанси до успіху.

3.3 Оцінка ефективності отриманої моделі

Створивши програму в Розділі 3.1 і маючи набір даних виконаних робіт (враховуючи методологію) і набір даних нового проєкту (без урахування методології) з Розділу 3.2 - проведено дослідження і підібрано методологію для нового проєкту, результати наведені в Таблиці 3.2

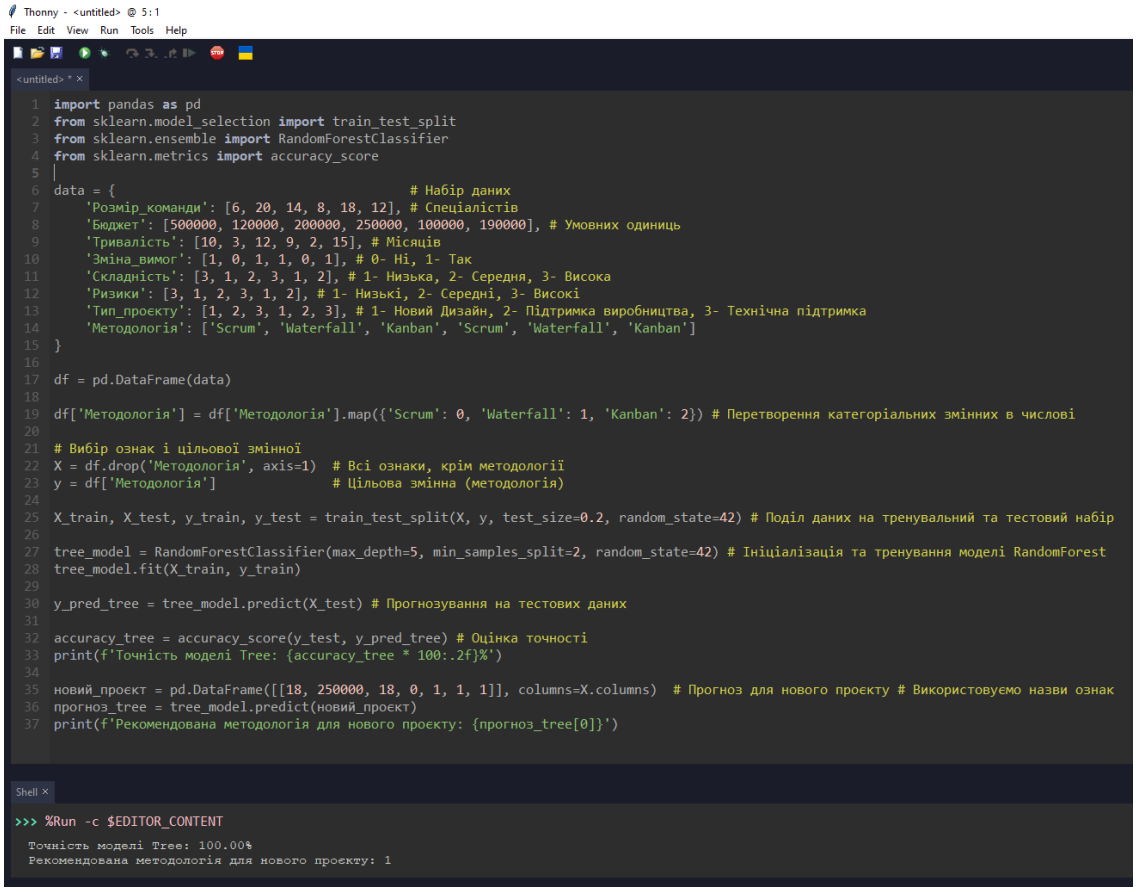


Рисунок 3.1 Програма для передбачення Методології нового проєкту / розроблено автором

Таблиця 3.2 Набір даних виконаних та нового проєкту з передбаченою Методологією за допомогою програми

Набір даних виконаних проєктів								
Шифр проєкту	*Розмір команди, чол.*	*Бюдж. тис. у.о.*	*Трив. міс.*	*Зміна вимог*	*Складність*	*Ризики*	*Тип проєкту*	*Метод.*
Проєкт 1	6	500	10	Так	Висока	Високі	Новий Дизайн	Scrum
Проєкт 2	20	120	3	Ні	Низька	Низькі	Підтримка виробництва	Waterfall
Проєкт 3	14	200	12	Так	Середня	Середні	Технічна підтримка	Kanban
Проєкт 4	8	250	9	Так	Висока	Високі	Новий Дизайн	Scrum
Проєкт 5	18	100	2	Ні	Низька	Низькі	Підтримка виробництва	Waterfall
Проєкт 6	12	190	15	Так	Середня	Середні	Технічна підтримка	Kanban
Набір даних нового проєкту								
Шифр проєкту	Розмір команди, чол.	Бюджет, тис. у.о.	Тривалість, міс.	Зміна вимог	Складність	Ризики	Тип проєкту	Підібрана метод.
Новий проєкт	18	250	18	Ні	Низька	Низькі	Підтримка виробництва	Waterfall

Модель Random Forest передбачає використання методології Waterfall (1). Це означає, що для проєктів підтримки виробництва, зі стабільними вимогами та без ризиків змін, модель вважає традиційний підхід Waterfall найбільш підходящим.

Для перевірки алгоритму та створеної програми на правильність підбору методологій – проведемо дослідження в якому використаємо набори даних п’яти виконаних проєктів для передбачення методолгії шостого виконаного проєкту. Вибірку проєктів виконаємо відповідно до методології таким чином щоб проєкт, що досліджується мав відмінну від попереднього дослідження методологію.

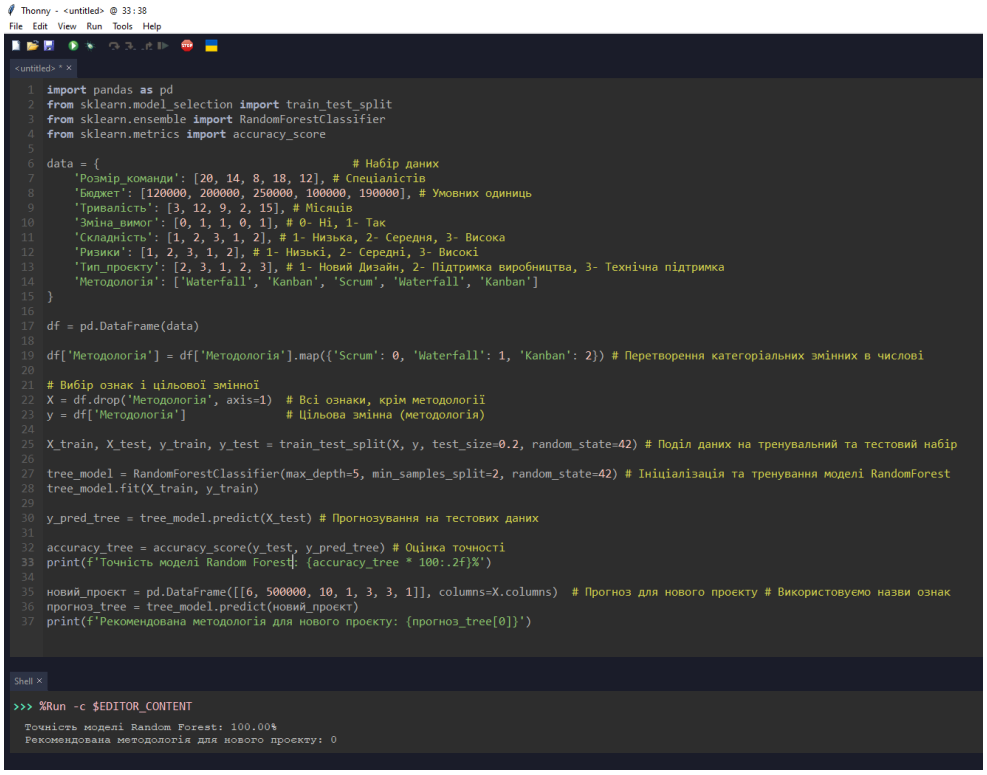


Рисунок 3.2 Програма для передбачення Методології проєкту котрий був виконаний за допомогою SCRUM / розроблено автором

Таблиця 3.3 Набір даних виконаних та нового проєкту з передбаченою Методологією за допомогою програми.

Набір даних виконаних проєктів								
Шифр проєкту	*Розмір команди, чол.*	*Бюджет, тис. у.о.*	*Тривалість міс.*	*Зміна вимог*	*Складність*	*Ризики*	*Тип проєкту*	*Методологія*
Проєкт 2	20	120	3	Ні	Низька	Низькі	Підтримка виробництва	Waterfall
Проєкт 3	14	200	12	Так	Середня	Середні	Технічна підтримка	Kanban
Проєкт 4	8	250	9	Так	Висока	Високі	Новий Дизайн	Scrum
Проєкт 5	18	100	2	Ні	Низька	Низькі	Підтримка виробництва	Waterfall
Проєкт 6	12	190	15	Так	Середня	Середні	Технічна підтримка	Kanban
Набір даних нового проєкту								
Шифр проєкту	Розмір команди, чол.	Бюджет, тис. у.о.	Тривалість, міс.	Зміна вимог	Складність	Ризики	Тип проєкту	Підібрана метод.
Проєкт 1	6	500	10	Так	Висока	Високі	Новий Дизайн	SCRUM

Модель Random Forest передбачає використання методології SCRUM (0) що співпало з обраною методологією.

Це означає, що для проєктів нового дизайну, зі гнучкими вимогами та високою складністю, модель вважає підхід Scrum найбільш підходящим.

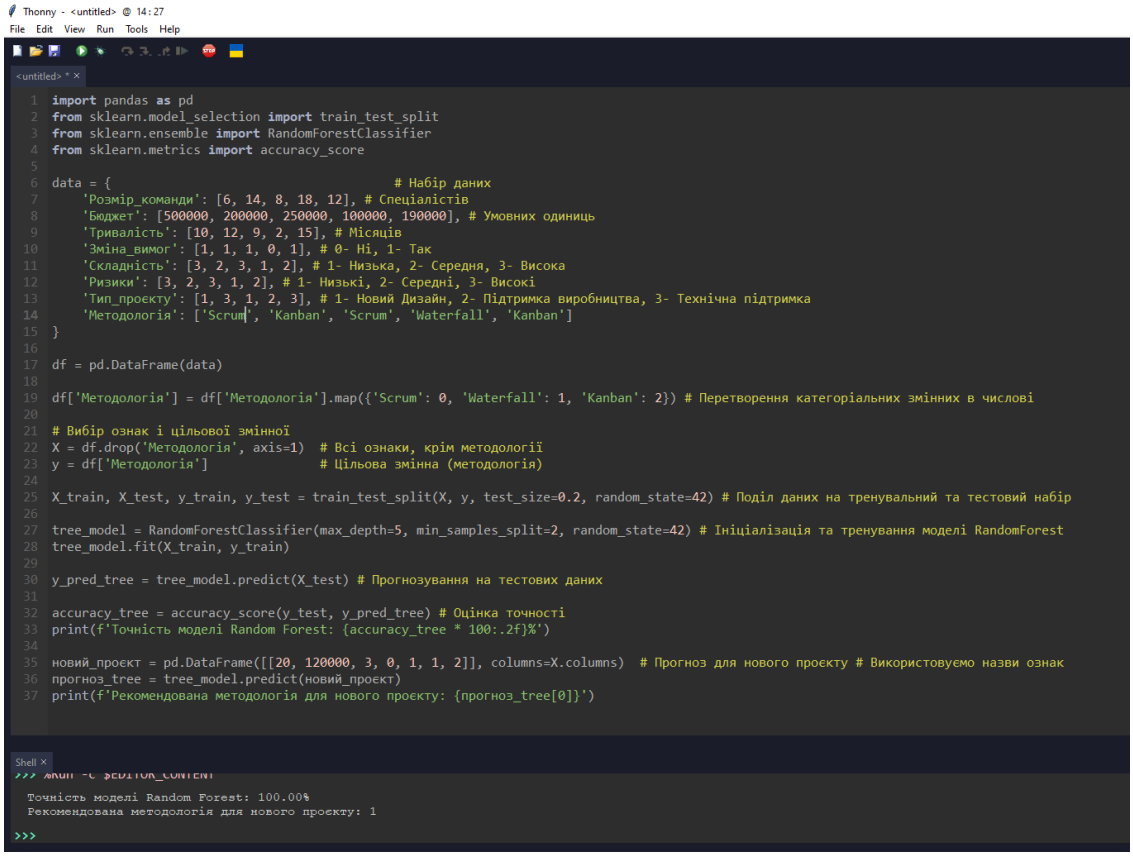


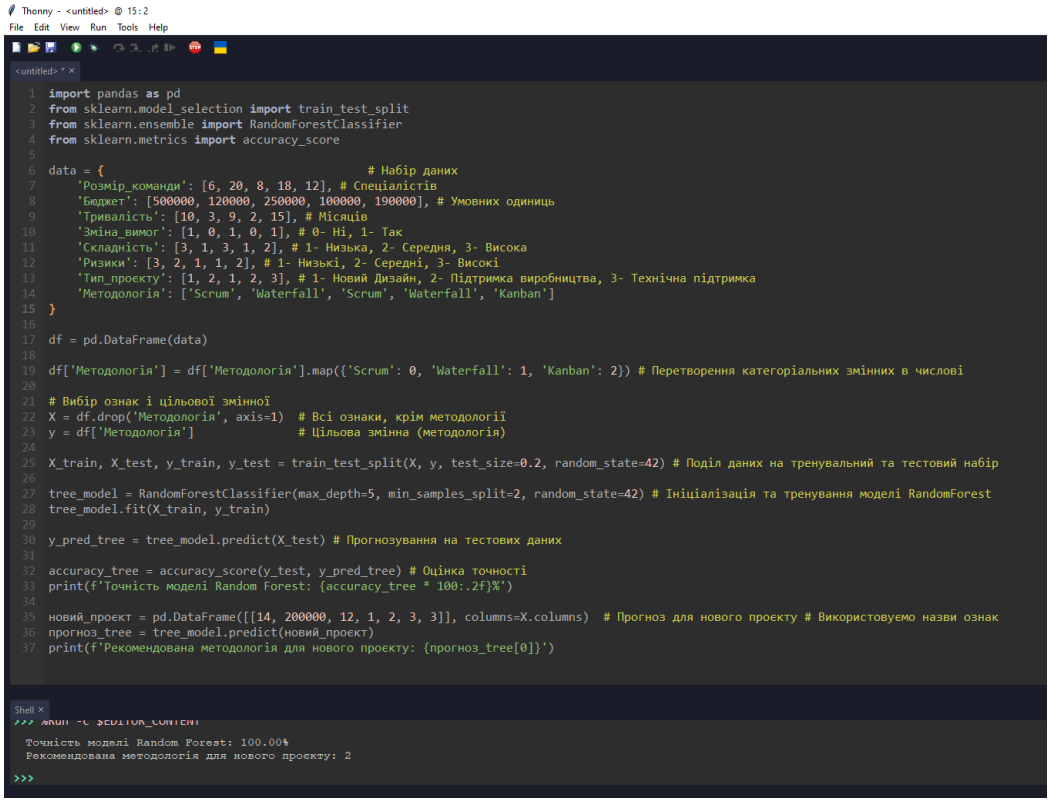
Рисунок 3.3 Програма для передбачення Методології нового проєкту котрий був виконаний за допомогою Waterfall / розроблено автором

Таблиця 3.4 Набір даних виконаних та нового проєкту з передбаченою Методологією за допомогою програми.

Набір даних виконаних проєктів								
Шифр проєкту	*Розмір команди, чол.*	*Бюджет, тис. у.о.*	*Тривалість міс.*	*Зміна вимог*	*Складність*	*Ризики*	*Тип проєкту*	*Метод.*
Проєкт 1	6	500	10	Так	Висока	Високі	Новий Дизайн	Scrum
Проєкт 3	14	200	12	Так	Середня	Середні	Технічна підтримка	Kanban
Проєкт 4	8	250	9	Так	Висока	Високі	Новий Дизайн	Scrum
Проєкт 5	18	100	2	Ні	Низька	Низькі	Підтримка виробництва	Waterfall
Проєкт 6	12	190	15	Так	Середня	Середні	Технічна підтримка	Kanban
Набір даних нового проєкту								
Шифр проєкту	Розмір команди, чол.	Бюджет, тис. у.о.	Тривалість, міс.	Зміна вимог	Складність	Ризики	Тип проєкту	Підібрана метод.
Проєкт 2	20	120	3	Ні	Низька	Низькі	Підтримка виробництва	Waterfall

Модель Random Forest передбачає використання методології Waterfall (1) що співпало з обраною методологією.

Це означає, що для проєктів підтримки виробництва, зі стабільними вимогами та без можливих ризиків , модель вважає традиційний підхід Waterfall найбільш підходящим.



```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

data = {                                    # Набір даних
    'Розмір_команди': [6, 20, 8, 18, 12], # Спеціалістів
    'Бюджет': [500000, 120000, 250000, 100000, 190000], # Умовних одиниць
    'Тривалість': [10, 3, 9, 2, 15], # Місяців
    'Зміна_вимог': [1, 0, 1, 0, 1], # 0- Ні, 1- Так
    'Складність': [3, 1, 3, 1, 2], # 1- Низька, 2- Середня, 3- Висока
    'Ризики': [3, 2, 1, 1, 2], # 1- Низькі, 2- Середні, 3- Високі
    'Тип_проєкту': [1, 2, 1, 2, 3], # 1- Новий Дизайн, 2- Підтримка виробництва, 3- Технічна підтримка
    'Методологія': ['Scrum', 'Waterfall', 'Scrum', 'Waterfall', 'Kanban']
}

df = pd.DataFrame(data)

df['Методологія'] = df['Методологія'].map({'Scrum': 0, 'Waterfall': 1, 'Kanban': 2}) # Перетворення категоріальних змінних в числові

# Вибір ознак і цільової змінної
X = df.drop('Методологія', axis=1)  # Всі ознаки, крім методології
y = df['Методологія']               # Цільова змінна (методологія)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # Поділ даних на тренувальний та тестовий набір

tree_model = RandomForestClassifier(max_depth=5, min_samples_split=2, random_state=42) # Ініціалізація та тренування моделі RandomForest
tree_model.fit(X_train, y_train)

y_pred_tree = tree_model.predict(X_test) # Прогнозування на тестових даних

accuracy_tree = accuracy_score(y_test, y_pred_tree) # Оцінка точності
print(f'Точність моделі Random Forest: {accuracy_tree * 100:.2f}%')

новий_проєкт = pd.DataFrame([[14, 200000, 12, 1, 2, 3, 3]], columns=X.columns)  # Прогноз для нового проєкту # Використовуємо назви ознак
прогноз_tree = tree_model.predict(новий_проєкт)
print(f'Рекомендована методологія для нового проєкту: {прогноз_tree[0]}')
```

```
Точність моделі Random Forest: 100.00%
Рекомендована методологія для нового проєкту: 2
>>>
```

Рисунок 3.4 Програма для передбачення Методології проєкту котрий був виконаний за допомогою KANBAN / розроблено автором

Таблиця 3.5 Набір даних виконаних та нового проєкту з передбаченою Методологією за допомогою програми.

Набір даних виконаних проєктів								
Шифр проєкту	*Розмір команди, чол.*	*Бюджет, тис. у.о.*	*Тривалість міс.*	*Зміна вимог*	*Складність*	*Ризики*	*Тип проєкту*	*Методологія*
Проєкт 1	6	500	10	Так	Висока	Високі	Новий Дизайн	Scrum
Проєкт 2	20	120	3	Ні	Низька	Низькі	Підтримка виробництва	Waterfall
Проєкт 4	8	250	9	Так	Висока	Високі	Новий Дизайн	Scrum
Проєкт 5	18	100	2	Ні	Низька	Низькі	Підтримка виробництва	Waterfall
Проєкт 6	12	190	15	Так	Середня	Середні	Технічна підтримка	Kanban
Набір даних нового проєкту								
Шифр проєкту	Розмір команди, чол.	Бюджет, тис. у.о.	Тривалість, міс.	Зміна вимог	Складність	Ризики	Тип проєкту	Підібрана метод.
Проєкт 3	14	200	12	Так	Середня	Середні	Технічна підтримка	Kanban

Модель Random Forest передбачає використання методології Kanban (2), що співпало з обраною методологією.

Це означає, що для проєктів технічної підтримки, зі стабільними вимогами та середньою складністю, модель вважає підхід Kanban найбільш підходящим.

Висновки до розділу 3

Дослідження у розділі 3 демонструє, що задані у гіпотезі показники, які використовувались у сеті даних дійсно впливають на вибір методології проєктного менеджменту. Дану модель рекомендується застосовувати для використання в дослідженнях, наукових проєктах або й на малих підприємствах котрі мають багато виконаних проєктів.

Для покращення роботи моделі необхідно розширити кількість вхідних показників, а також додати дані з інших підприємств, таким чином підвищивши точність моделі.

ВИСНОВКИ

Управління ІТ-проєктами лежить у площині загальної теорії проєктного менеджменту, однак вирізняється постійною динамікою розвитку і мінливості.

Перший розділ дипломної роботи присвячений теоретичним основам проєктного менеджменту. Методології управління проєктами, зокрема у ІТгалузі визначають шлях проєкту і принципи роботи проєктної команди, задають напрям взаємодії всередині команди і спілкуванню з замовником. Правильно вибрана методологія підвищує шанси проєкту на успішність, що узагальнено можна виміряти класичним трикутником проєктного менеджменту:

- чи був закінчений проєкт вчасно;
- чи було дотримано заданого кошторису;
- и задовольняє якість та функціонал очікування бізнесу.

Гнучкі та каскадні групи методологій є основними для сучасногопроєктного менеджменту, що зумовленою простотою їх трактування і застосування. Кожна група методологій відповідає окремим вимогам і потребам кінцевого продукту, а також моделі життєвого циклу програмного забезпечення. Найпопулярніші методології проєктного менеджменту – Scrum, Kanban і Waterfall. Scrum вирізняється поступовим нарощуванням функціоналу продукту через постійні проміжні релізи, Kanban підходить для безперервних процесів, у яких мало уваги приділяється плануванню, а Waterfall має чітко визначені вимоги і фіксовані етапи розробки.

Виявилося, що проблема визначення правильної методології проєктного менеджменту є однією із ключових задач проєктної діяльності, адже організація процесів відповідно до методології значним чином впливає на результат проєкту. Неправильно підібрана методологія може спричинити репутаційні втрати – через непорозуміння з замовниками або інвесторами, фінансові втрати – через некоректність визначення вимог, перевищення

терміну розробки проєкту – через недостатній аналіз та неефективну взаємодію у команді. За вибір методології управління проєктами зазвичай відповідальний проєктний менеджер або операційний директор підприємства. Для довгострокових проєктів це може стати стратегічним рішенням, яке закладе підвалини майбутньої взаємодії усіх зацікавлених сторін. Цей процес доволі одноманітний, тому може бути оптимізований з використанням машинного алгоритму, що полегшить роботу проєктного менеджера і дозволить зосередитися на інших аспектах запуску проєкту.

У другому розділі увагу зосереджено на виборі алгоритмів машинного навчання, прикладних випадках застосування нейронних мереж в управлінні проєктами та аналізі вхідних даних. Коли йдеться про вибір методології, мається на увазі чітко визначена категорія, яка підпадає під визначення задачі класифікації. Оптимальним рішенням для цього завдання є застосування нейронної мережі. Нейронна мережа може визначити і передбачити приховані закономірності, чим полегшить роботу проєктного менеджера. Вхідні дані повинні відповідати чітким критеріям, для того, щоб нейронна мережа могла видати результат із високою точністю. Тому можливе використання виключно кількісних характеристик проєкту.

У третьому розділі проводиться дослідження метою якого є перевірка можливості автоматизації управління методологією IT-проєкту. Кожен проєкт має категорію методології: Scrum, Kanban або Waterfall. Така модель може бути легко впроваджена на підприємстві. Для цього не потрібно залучати додаткові фінансові ресурси. Однак спершу необхідно глибше проаналізувати усі характеристики проєктів на підприємстві і визначити найуспішніші з них.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ноздріна Л. В., Ящук В. І., Полотай О. І. Управління проектами: Підручник / За заг. ред. Л. В. Ноздріної. — К.: Центр учбової літератури, 2010. — 432 с.

2. Femi A. Benefits of using Pandas [Електронний ресурс] / Anthony Femi // Packt Publishing Ltd.. – 2015. – Режим доступу до ресурсу: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781783 981960/1/ch01lvl1sec11/benefits-of-using-pandas.

3. Катренко А.В. Управління ІТ-проектами . [Книга 1. Стандарти, моделі та методи управління проектами] : [підручник]. – Львів: «Новий Світ 2000» , 2013. – 550 с.

4. Кальніченко О. В., Чернова М. Л. Розробка та впровадження інформаційної системи управління проектами в будівельних компаніях //Технологический аудит и резервы производства — № 2/2(22), 2015. – С. 54–58.

5. Hasan S. Top 20 Project Management Methodologies & their Scions: Which One You Should Choose & Why? [Електронний ресурс] / Sarmad Hasan // Taskque. – 2017. – Режим доступу до ресурсу: https://blog.taskque.com/topproject-management-methodologies.

6. Chin, C. M. M., Spowage, A. C. 2012. Project Management Methodologies: A Comparative Analysis. Journal for the Advancement of Performance Information and Value, 4 (1), 106-118.

7. Міжнародний стандарт з управління проектами ISO 21500:2012 [Електронний ресурс] // Міжнародна організація зі стандартизації ISO. – 2012. – Режим доступу до ресурсу: https://www.iso.org/.

8. Neural networks application for automation of the IT project management methodology implementation / M. Ivashchenko, O. Mezentseva. // III

International Scientific and Practical Conference "Innovative development of Science and education". – Athens, 2020.

9. What Is Scrumban? How It Differs from Scrum & Kanban [Електронний ресурс] // ProjectManager. – 2020. – Режим доступу до ресурсу: https://www.projectmanager.com/blog/what-is-scrumban.

10. Калініченко Л. Л. Формування та оцінювання ефективності проектного менеджменту / Людмила Леонідівна Калініченко. // Маркетинг і менеджмент інновацій. – 2016. – №4. – С. 169–179.

11. SDLC (Software Development Life Cycle) Tutorial: What is, Phases, Model [Електронний ресурс] // Guru99. – 2015. – Режим доступу до ресурсу: https://www.guru99.com/software-development-life-cycle-tutorial.html.

12. Trask, A.: Grokking Deep Learning. Print Inc., USA. 310p. (2019).

13. Big Bang Model [Електронний ресурс] // javaTpoint. – 2017. – Режим доступу до ресурсу: https://www.javatpoint.com/software-engineering-big-bang-model.

14. T. Cooke-Davies, “The ‘real’ success factors on projects,” Int. J. Proj. Manag., vol. 20, no. 3, pp. 185–190, Apr. 2002.

15. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. & Thomas, D. (2001). Manifesto for Agile Software Development

16. Табунщик Г.В Інженерія якості програмного забезпечення:навчальний посібник / Г.В Табунщик, Р.К. Кудерметов, Т.І. Брагіна. – Запоріжжя: ЗНТУ, 2013. – 180 с.

17. Agile Project Management: Best Practices and Methodologies [Електронний ресурс] // AltexSoft. – 2018. – Режим доступу до ресурсу: https://www.altexsoft.com/whitepapers/agile-project-management-best-practicesand-methodologies/.

18. Vargas, R. V. (2015). Applying neural networks and analogous estimating to determine the project budget. Paper presented at PMI® Global

Congress 2015—North America, Orlando, FL. Newtown Square, PA: Project Management Institute.

19. What Is Scrumban? How It Differs from Scrum & Kanban [Електронний ресурс] // ProjectManager. – 2020. – Режим доступу до ресурсу: https://www.projectmanager.com/blog/what-is-scrumban.

20. What is a random forest? – Режим доступу до ресурсу: https://www.spotfire.com/glossary/what-is-a-random-forest