Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет_____ Комп'ютерних наук_____
(повна назва)

Кафедра _____Інформаційних управляючих систем_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

_____ Дослідження методів підтримки прийняття рішень при _____
_____удосконаленні процесу розробки ІТ-проекту_____.
(тема)

Виконав:
студент __2__ курсу, групи ___УПГІТм-20-1__
_____Потєхін С. В._____
(прізвище, ініціали)

Спеціальність __122 Комп'ютерні науки_____
_____
(код і повна назва спеціальності)

Тип програми __освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма____Управління проектами в галузі інформаційних технологій
_____
(повна назва освітньої програми)

Керівник__проф. Чалий С. Ф._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ ___Петров К.Е.____
(підпис) (прізвище, ініціали)

2021р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерних наук_____
(повна назва)

Кафедра_____Інформаційних управляючих систем_____
(повна назва)

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність _____122 Комп'ютерні науки_____
(код і повна назва)

Тип програми __освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма__Управління проектами в галузі інформаційних технологій__
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові_____Потєхіну Севастіану Валеріановичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Дослідження методів підтримки прийняття рішень при удосконаленні процесу розробки ІТ-проекту_____
_____
затверджена наказом по університету від ____05.11_____2021 р. № ___1646 Cm_____
2. Термін подання студентом роботи до екзаменаційної комісії ____13.12_____ 2021_ р.
3. Вихідні дані до роботи _Науково-технічні публікації та інтернет джерела з тематики__ кваліфікаційної роботи_____
_____
_____
_____
_____
_____
4. Перелік питань, що потрібно опрацювати в роботі _вступ, аналіз процесу розробки ІТ проекту, структуризація процесу підтримки прийняття рішень щодо удосконалення процесу розробки ІТ проекту, удосконалення методу визначення пріоритетів розробки задач ІТ проекту з використанням моделі задоволення потреб Кано, планування проекту розробки методу визначення пріоритетів розробки задач ІТ проекту, експериментальна перевірка отриманого удосконаленого методу визначення пріоритетів розробки задач ІТ проекту, висновки._____
_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз літератури та Інтернет-джерел	08.11.2021	
2	Постановка задачі	09.11.2021-11.11.2021	
3	Обробка матеріалу	12.11.2021-13.11.2021	
4	Дослідження процесу розробки ІТ проекту	14.11.2021-15.11.2021	
5	Дослідження методів методів визначення пріоритетів розробки задач ІТ проекту	15.11.2021-16.11.2021	
6	Дослідження особливостей методів реалізації визначення пріоритетів розробки задач ІТ проекту	17.11.2021-18.11.2021	
7	Апробація результатів дослідження на прикладі	19.11.2021-20.11.2021	
8	Написання пояснювальної записки	21.11.2021-28.11.2021	
9	Підготовка презентації	29.11.2021-02.12.2021	
10	Перевірка на плагіат	03.12.2021-06.12.2021	
11	Нормоконтроль	07.12.2021-10.12.2021	
12	Захист	14.12.2021	

Дата видачі завдання _____08.11_____2021 р.

Студент _____
(підпис)

Керівник роботи _____ _____проф. Чалий С. Ф_____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи містить: 84 с., 4 розділи, 11 рис., 3 табл., 28 джерел.

ВИЗНАЧЕННЯ ПРІОРИТЕТУ, ДІАГРАМА ГАНТА, ІТЕРАЦІЯ, МЕТОДОЛОГІЯ РОЗРОБКИ, ПІДТРИМКА ПРИЙНЯТТЯ РІШЕНЬ, РОЗРОБКА ІТ ПРОЕКТУ, AGILE МЕТОДОЛОГІЯ, SCRUM, USER STORY.

У роботі проведено дослідження проблеми підтримки удосконалення процесу розробки ІТ проекту. вирішено задачі структуризації процесу підтримки прийняття рішень щодо удосконалення процесу розробки ІТ проекту; удосконалення методу визначення пріоритетів розробки задач ІТ проекту з використанням моделі задоволення потреб Кано; планування проекту розробки методу визначення пріоритетів розробки задач ІТ проекту; експериментальної перевірки отриманого удосконаленого методу визначення пріоритетів розробки задач ІТ проекту. В ході дослідження удосконалено метод визначення пріоритетів розробки задач ІТ проекту з використанням моделі задоволення потреб Кано шляхом уточнення пріоритетів на основі різниці балів користувачів щодо наявності та відсутності таких задач у проекті. Метод додатково враховує бали користувачів щодо задач та затримки при розробці проекту, що дає можливість динамічно змінювати процес розробки на основі уточнення послідовності реалізації задач.

ABSTRACT

Explanatory Note to certification work contains 84 pages, 4 sections, 11 pictures, 2 tables, 28 sources.

PRIORITY DETERMINATION, GANT DIAGRAM, ITERATION, DEVELOPMENT METHODOLOGY, DECISION SUPPORTING, IT PROJECT DEVELOPMENT, AGILE USE METHOD.

The study of the problem of supporting the improvement of the IT project development process was conducted. solved the problem of structuring the decision support process to improve the process of IT project development; improving the method of determining priorities for the development of IT project tasks using the model of meeting the needs of Kano; planning of the project of development of a method of definition of priorities of development of tasks of the IT project; experimental verification of the obtained improved method of determining the priorities of development of IT project tasks. The study improved the method of prioritizing the development of IT project tasks using the Kano needs satisfaction model by refining priorities based on the difference in user scores on the presence and absence of such tasks in the project. The method additionally considers users' scores on tasks and delays in project development, which allows to dynamically change the development process based on the specification of the sequence of tasks.

ПЕРЕЛІК СКОРОЧЕНЬ, УСЛОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ТА ТЕРМІНІВ

СППР– системи підтримки прийняття рішень;

XP – екстремальне програмування

Scrum/XP комбіноване екстремальне програмування з використанням методології Scrum ;

JRE – Java Runtime Enviroment.

JVM– Java Virtual Machine.

ЗМІСТ

ВСТУП

Процес розробки IT проекту орієнтований на вирішенням проблем, пов’язаних з технічними та управлінськими аспектами з розробки програмного забезпечення ІС. IT-проект охоплює як діяльність з розробки, так і діяльність з управління та удосконалення процесу розробки. Фундаментальні цілі даного процесу пов'язані із оптимальністю та масштабованістю програмних проектів.

Оптимальність означає, що процес розробки IT проекту повинен забезпечувати можливість створення високоякісних програмних систем за низькою вартістю, а масштабованість означає, що він також має бути застосовним для великих IT проектів. Для досягнення цих цілей процес має бути передбачуваним. Передбачуваність процесу визначає, наскільки точно можна прогнозувати результат виконання IT проекту до його завершення. Передбачуваність можна вважати фундаментальною властивістю будь-якого процесу.

Метою процесу розробки IT проекту має бути не лише скорочення зусиль при проектуванні та кодуванні, а й зниження витрат на обслуговування. Отже, на ранніх етапах процесу розробки головними питаннями повинні бути «чи можна легко перевірити» і «чи можна легко змінити». Помилки можуть виникнути на будь-якому етапі розробки. Однак виявлення та виправлення помилок має бути безперервним процесом, який виконується протягом усієї розробки IT проекту. Процес розробки є також не є статичною сутністю.

Оскільки продуктивність (і, отже, вартість проекту) і його якість визначаються значною мірою процесом розробки, то для задоволення інженерних цілей покращення якості та зниження витрат, процес розробки IT проекту необхідно покращувати. Тому вдосконалення процесу є його важливою метою.

Сучасний стан управління IT проектами характеризується постійним удосконаленням на основі переходу від традиційного планового управління проектами до agile - стилю управління, що керується подіями.

Agile-менеджмент проектів деякий час розглядався як новий етап розвитку процесів розробки IT проектів, який зробить революцію в індустрії програмного забезпечення. На сьогодні agile є новим стандартом для проектів розробки програмного забезпечення. Концепція agile існує вже деякий час, і хоча знання та практики використання методології розширюються, agile не завжди дає можливість ефективно розробляти проекти. Розуміння того, коли використовувати agile і які фактори успіху враховувати, важливо для успішної розробки IT проектів.

На сьогодні у парадигмі agile акцент робиться на прийнятті рішень із удосконалення процесу розробки, на уточненні вимог клієнтів, тощо.

Для вирішення проблеми постійного удосконалення процесу розробки IT використовується ітераційна модель життєвого циклу проекту. Вказана модель передбачає послідовне виконання фаз визначення області застосування, планування, запуску, моніторингу результатів та переходу до нової ітерації.

Удосконалення процесу розробки базується на використанні існуючих підходів до підтримки прийняття рішень. Виділяють групи раціональних та обмежено раціональних підходів. Раціональний підхід потребує повної інформації щодо предметної області. Така властивість приводить до труднощів при його практичному використанні. Тому на практиці при розробці IT проектів знайшли застосування переважно обмежено раціональні підходи. Вони базуються на обмеженні вхідних даних та виборі невеликої кількості найбільш сприятливих альтернатив.

Послідовність підтримки прийняття рішень включає в себе етапи виявлення проблем у процесі розробки IT проекту, а також побудови та оцінки альтернативних рішень із удосконалення даного процесу.

Виявлення проблем у процесі розробки ІТ проекту за гнучкою методологією відбувається в першу чергу шляхом проведення процедури ретроспективи. Остання містить виявлення сильних та слабких сторін і обмежень процесу розробки та подальше уточнення рішень із удосконалення такого процесу.

Побудова рішень із удосконалення процесу значною мірою пов'язана із обґрунтованою зміною пріоритетів задач ІТ проекту. Для підтримки визначення пріоритетів використовуються підходи, що основані на строках виконання задач, потребах користувачів, характеристиках задач та вартості затримок їх виконання. Однак необхідність постійного удосконалення процесу розробки при зміні оцінок користувачів, затримках у розробці задач потребують уточнення пріоритетів на основі інтеграції даних підходів, що показує актуальність теми роботи.

Об'єктом даного дослідження є процес підтримки прийняття рішень при розробці ІТ- проекту.

Предметом даного дослідження є методи підтримки прийняття рішень при удосконаленні процесу розробки ІТ проекту.

Метою роботи є дослідження методів підтримки прийняття рішень у процесі у розробки ІТ проекту для задоволення потреб користувачів продукту, що розробляється.

1 АНАЛІЗ ПРОЦЕСУ РОЗРОБКИ ІТ ПРОЕКТУ

1.1 Аналіз структури процесу розробки ІТ проекту

Розробка підходів до управління ІТ проектами почалась у кінці 1960-х років. Індустрія програмного забезпечення швидко розвивалась, і комп'ютерні компанії побачили потенціал у створенням програмного забезпечення, що мало низьку вартість порівняно з виробництвом апаратного забезпечення та схем, які були більш поширеними на той час.

Компанії, що займаються розробкою програмного забезпечення, використали відому водоспадну модель при розробці програмних проектів. Waterfall – це модель розробки проекту, яка використовується в багатьох галузях промисловості. Це була перша модель проекту, яка використовувалася в життєвому циклі створення програмного забезпечення. Водоспадна модель дає покроковий план роботи, де кожен крок має бути виконаний, перш ніж переходити до наступного. Реалізується лінійна модель послідовного життєвого циклу. Водоспад починається зі збору команди та документування вимог. Під час цього процесу клієнти та зацікавлені сторони мають детально описати вимоги. Далі розробники проектують рішення, яке відповідає вимогам. Далі вони пишуть код і тестують. Нарешті, команда виправляє будь-які помилки або неточності та передає готовий проект замовнику. Команда розроблювачів програмного забезпечення зобов'язана закінчити кожен крок, перш ніж почати наступний.

У моделі водоспаду є шість фаз, які представлені на рис1.1. Кожна з фаз розроблена так, щоб інтуїтивно перетікати в наступну.

На першій фазі команда проекту отримує всі вимоги від клієнта. Вони повинні бути дуже детальними з урахуванням чіткого бачення кінцевого продукту. Вимоги включають його функції, призначення та унікальні характеристики. В подальшому вимоги не змінюються.

На фазі проектування програмної системи архітектор разом із членами команди створюють проект для програмного забезпечення. Після визначення специфікацій обладнання та програмного забезпечення вони пишуть весь необхідний код.

Після того, як проект підготовлений, розробники створюють програмне забезпечення із окремих модулів. Кожен модуль розробляється окремо, а потім перевіряється в рамках тестування. Кожен модуль має бути повністю працездатним перед наступною фазою.

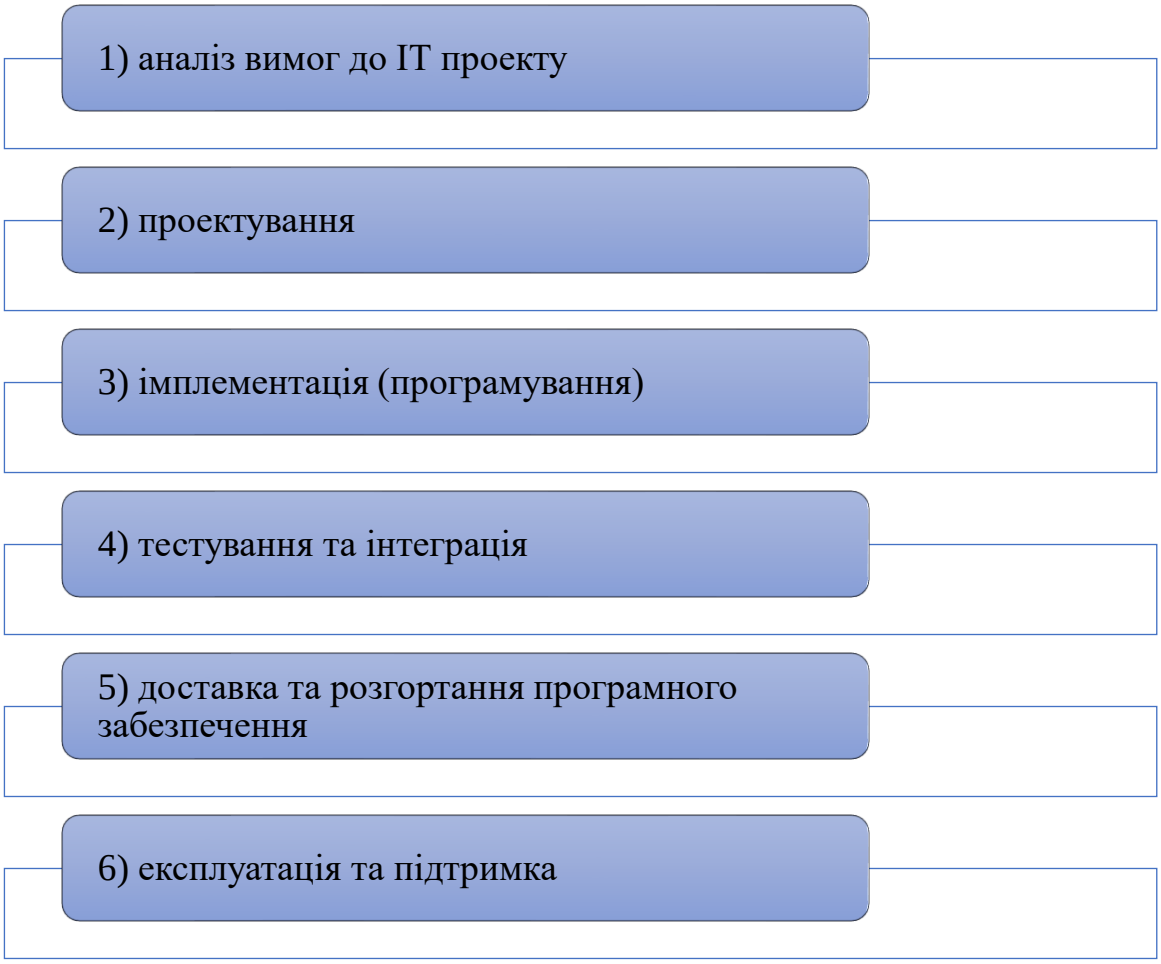

Рисунок 1.1 – Водоспадна модель

Протягом виконання фази тестування та інтеграції модулі проекту об’єднуються в систему, яка в кінцевому підсумку стане кінцевим продуктом.

Команда тестує систему, переконавшись, що всі підрозділи працюють разом належним чином. Будь-які дефекти усуваються в міру їх виникнення.

Після побудови системи, на фазі доставки та розгортання, команда встановлює систему, запускає тести на серверах, а потім програмне забезпечення починає використовуватися. Під час цього початкового періоду запуску команда перебуває в режимі очікування, щоб контролювати функціональність програмного забезпечення.

Після запуску програмного забезпечення розробники можуть вносити зміни або інші незначні зміни за запитом клієнта. Усі дефекти, які виникли або були упущені на попередніх етапах, також усуваються. З часом клієнту надається регулярна підтримка та технічне обслуговування.

Водоспадна модель має такі переваги:

– пропонує обширну документацію; усі етапи проекту мають детальну документацію для усунення непорозумінь;

– команди знають обсяг роботи завдяки ретельному збору вимог

– залучення клієнтів не обов’язкове під час проекту, лише на початку та в кінці.

З часом виявилося, що такий поступовий підхід до розробки програмних систем не є оптимальним. Такий підхід має наступні обмеження:

– жорстко заданий поділ фаз розробки;

– вимоги не завжди зрозумілі на початку проекту, та можуть змінюватись у процесі його розвитку

Ці обмеження приводили до збоїв у розробці ІТ проектів внаслідок таких причин [1-3]:

– нереалістичні цілі проекту;

– неточні оцінки проекту;

– неточно визначені вимоги;

– погана звітність про стан розробки;

– некеровані ризики;

– погана комунікація із замовником;

– використання незрілих технологій написання коду;

– висока складність проекту;

– відсутність практики розвитку проекту;

– неефективне управління внаслідок політики зацікавлених сторін;

– комерційний тиск.

Такі недоліки потребували нових підходів, і кілька нових «полегшених» методів розробки IT проектів почали з’являтися в кінці 1980-х і протягом 1990-х років. Методи використовували ітераційний і поетапний підхід до розробки. Це було покликано сприяти тіснішій співпраці із замовником, заохочуючи зміни, що відбуваються протягом усього проекту, щоб краще задовольнити потреби клієнтів.

У 2001 році представники кількох найбільш важливих легких методологій сформували Agile Alliance. Був створений Маніфест для Agile Software Development (Agile Alliance) [4-7]. Вони визначили чотири базових цінності Agile, яким повинні відповідати всі Agile-методології (рис. 1.2).

Ці елементи контрастують із традиційним підходом, орієнтованим на плани. Дотримання цих цінностей і принципів має на меті збільшити ймовірність успіху розробки IT проекту.

Найбільш актуальні відмінності між водоспадною моделлю та методологією Agile наведено в табл 1.1.

На практиці кількість невдалих або оскаржуваних програмних проектів є досить великою. У роботі [8,9] повідомляється, що лише 39 відсотків усіх проектів можна класифікувати як успішні. 43 відсотки були оскаржені (виконані запізно, перевищено бюджет та/або відсутні функції), а 18 відсотків – невдалими (скасовані або доставлені, але ніколи не використані). Відзначається поступове збільшення показників успішності з 29 відсотків у 2004 році.

У звіті також вказується, що розмір проекту важливіший за те, чи є використана методологія швидкою чи традиційною. Вони стверджують, що причиною цього є те, що гнучкі методології полегшують створення невеликих

проектів, а великий проект в 10 разів частіше стикається з проблемами в порівнянні з маленьким проектом.

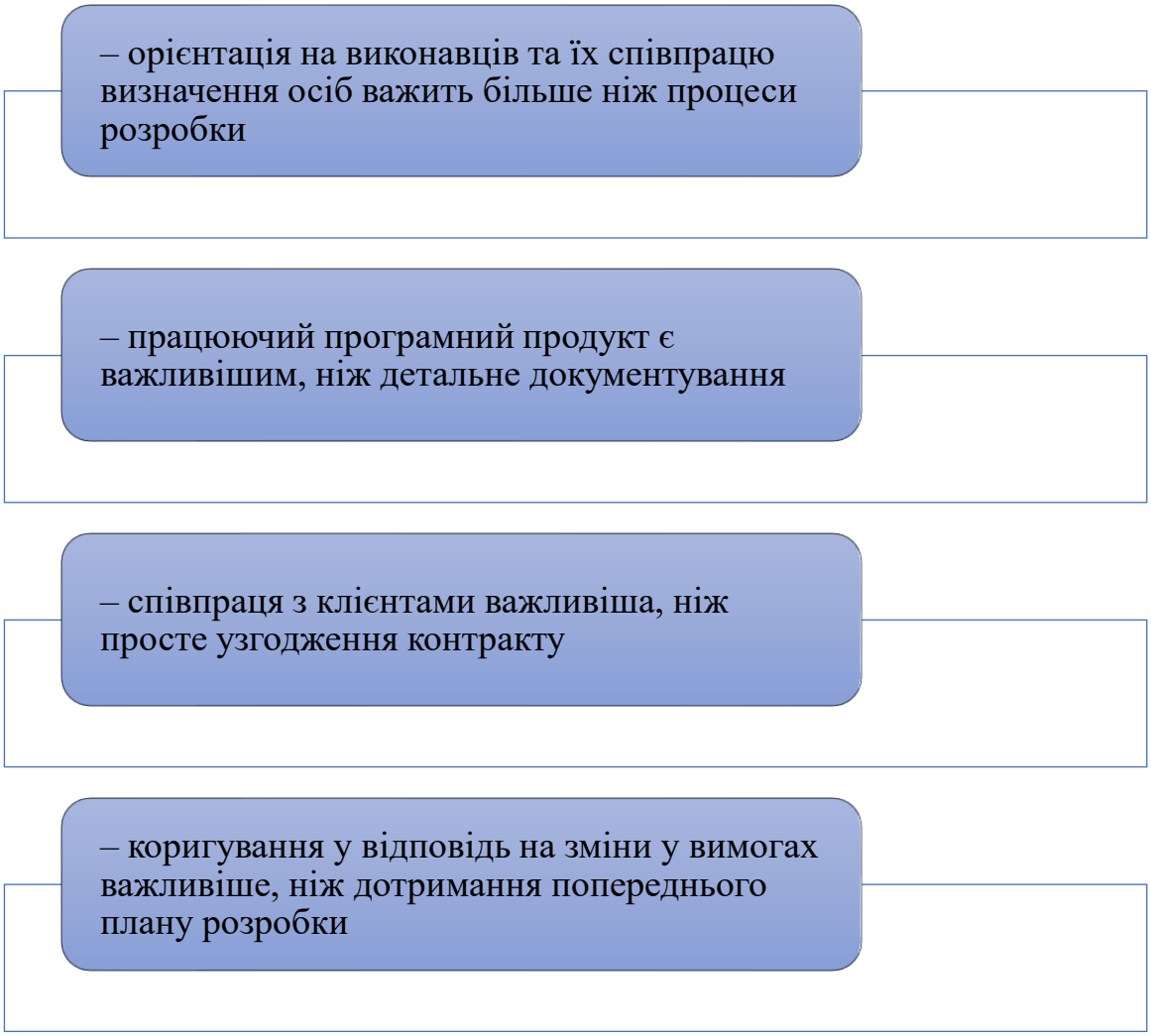

Рисунок 1.2 – Цінності Agile-методології

На сьогодні використання agile розширюється. Так, 45 відсотків респондентів використовують agile у більшості своїх проектів. 90 відсотків організацій, які брали участь у анкетуванні, стверджують, що використовують гнучкі розробки. Scrum – це найкраща методологія, яку використовують 56 відсотків agile-команд. Якщо включити різні гібриди Scrum, це число збільшується до 72 відсотків [10].

Таблиця 1.1 – Порівняння водоспадної моделі та Agile

Водоспадна модель	Agile
Потребує точного та довгострокового планування, вимоги зазвичай статичні	Короткострокове планування із динамічним доповнення вимог
Ієрархічна структуру команди	Кожен член команди грає однакову роль
Не залежить від спілкування з клієнтом	Постійне спілкування з клієнтом
Контроль якості може бути ускладненим через пізнє тестування	Контроль якості перевіряється шляхом послідовного тестування
Прогрес виконання вимірюється за реалізованими кроками	Прогрес вимірюється за виконанням одиниці або спринту
Терміни можна чітко визначити, якщо не виникне серйозних проблем	Терміни поставки можуть змінитися, якщо зміняться вимоги
Краще підходить для коротких, простіших проектів, для яких відомі всі вимоги	Проекти можуть бути складними, поточними та довгостроковими

У сфері розробки програмних систем поширена точка зору, що при використанні agile зростає ймовірність успіху програмного проекту. Незважаючи на те, що це не завжди підтверджується емпіричними даними, agile деякий час сприймалась як технологія, що зробить революцію в розробці програмного забезпечення та програмних проектах. Недавнє дослідження успіху Agile проектів свідчить про можливість такого твердження. В роботі [7, 11-13] провели кількісне дослідження, щоб перевірити, чи впливає використання agile методів на успіх IT-проекту. Вони виявили ознаки того, що використання гнучких методів корелює з більш високим зареєстрованим

показником успіху. Це було показано для трьох категорій успіху; загальний успіх проекту, ефективність та успіх зацікавлених сторін.

Agile- проекти характеризуються більш гнучким процесом у порівнянні з традиційними проектами. Цей процес складається з набору практик, який описує процедури, які використовує команда розробників для досягнення цілей проекту.

Для IT проектів , які є проектами, метою яких є створення робочого програмного продукту, цей гнучкий підхід допомагає визначити масштаб проекту протягом усього його життя. Обсяг, а також час, вартість та якість є важливими критеріями при розгляді успіху програмного проекту.

1.2 Дослідження моделей життєвого циклу управління проектами

Проект – це обмежена у часі діяльність, результатом якої є унікальний продукт, послуга чи результат [14, 15]. Проект має чітко визначений початок і кінець. Кінцевий стан досягається, коли цілі проекту досягнуті або проект припинено з інших причин.

Управління проектами визначається як «застосування знань, навичок, інструментів і методів до проектної діяльності для задоволення вимог проекту» [16, 17]. Відповідно до посібника PMBOK, управління проектами складається з п'яти груп процесів [18]:

– ініціація;

– планування;

– виконання;

– моніторинг;

– контроль;

– закриття.

У традиційному управлінні проектами цей процес виконується лінійно та поступово. При гнучкому управлінні це робиться більш ітеративним і адаптивним способом.

Виходячи з цих груп процесів, визначаються п’ять різних моделей процесів життєвого циклу управління проектами, ак саме [19, 20]:

– лінійна;

– інкрементна;

– ітераційна;

– адаптивна;

– екстремальна.

Рішення про те, яку модель життєвого циклу вибрати, залежить від ступеня невизначеності мети проекту.

Лінійна і інкрементальна моделі реалізують традиційний цикл управління проектами. Вони мають низький рівень невизначеності рішення, а мета проекту зрозуміла.

Ітеративна та адаптивна моделі реалізують гнучкий життєвий цикл. Вони мають високий рівень невизначеності щодо рішення, але мета проекту все ще є зрозумілою.

Екстремальна модель відповідає екстремальному управлінню проектами. Екстремальний життєвий цикл має високий рівень невизначеності щодо рішення, а мета проекту є не повністю ясною. Або ж невизначеність рішення низька, але мета проекту неясна. Таблиця 1.2 ілюструє різницю в моделях життєвого цикла.

У лінійній моделі кожна група процесів виконується рівно один раз, послідовно. Ця модель не підтримує зміни обсягу вимог. Продукт поставляється відповідно до вимог, які був задані на початку проекту.

Інкрементна модель аналогічна лінійній моделі, але частину вимог можна доповнити. Це дозволяє отримати кращий зворотний зв’язок від клієнтів та заохочує до удосконалення вимог. Перевага, порівняно з лінійною

моделлю, полягає в тому, що кінцевий продукт буде ближче до того, що потребує клієнт.

Таким чином, традиційний підхід є підходом, керованим планом.

Таблиця 1.2 – Порівняння можливостей застосування моделей життєвого циклу IT проекту

Модель життєвого циклу	Технологія управління	Можливості застосування	
		Невизначеність щодо рішення	Зрозумілість мети проекту
Лінійна	Водоспадна	Низький рівень невизначеності	Так
Інкрементна	Водоспадна	Низький рівень невизначеності	Так
Ітераційна	Гнучка	Високий рівень невизначеності	Так
Адаптивна	Гнучка	Високий рівень невизначеності	Так
Екстремальна	Гнучка	Високий рівень невизначеності	Мета проекту є не повністю зрозумілою

Коли мета зрозуміла, але незрозуміло рішення і шлях її досягнення, використовується Agile -підхід. Ці типи проектів визначаються як складні та вимагають нетрадиційного підходу для успішного виконання проекту.

Навпаки, проекти Agile керуються змінами. Це означає, що замість того, щоб уникати змін у проекті, зміни заохочуються. Це спричиняє більш динамічні відносини між командою, яка створює продукт, та зовнішніми зацікавленими сторонами, які його попросили. Зацікавлені сторони більше

залучаються до процесу, але кінцевий результат буде набагато ближче до того, чого вони насправді бажають.

Оскільки кінцеве рішення та обсяг робіт є незрозумілими на початку, вони визначається протягом усього проекту на основі відгуків замовника та інших зацікавлених сторін.

Ітераційна модель покращує покрокову модель, включаючи планування в циклі реалізації. Кожен цикл створює потенційний код для використання замовником (може бути запущений у виробництво), про який зацікавлені сторони можуть дати відгук. Зворотній зв'язок вітається і є обов'язковим компонентом процесу. На основі відгуків і загального бачення продукту планується, виконується та впроваджується нова ітерація. Таким чином, рішення не повністю відоме на початку, але визначається протягом усього життя проекту.

На рис. 1.3 представлено послідовність кроків в ітераційній моделі.

Рисунок 1.3 – Послідовність кроків ітераційної моделі

Agile методи бувають як інкрементними, так і ітеративними . Інкрементний, оскільки робота попередньо поділено на менші пакети робіт, і ітераційний, оскільки обсяг кожного пакету визначається безпосередньо перед початком кожного циклу. Цей ітеративний характер робить процес дуже гнучким.

Адаптивна модель дуже схожа на ітераційну модель, але вона має ще менший час циклу, що полегшує реагування на зміни вимог. Основна відмінність між адаптивною та ітераційною моделлю полягає в тому, що більша частина рішення невідома в адаптивній моделі. Чим менше буде відомо, тим більше буде ризиків і складності. Коли складність висока, адаптивна модель є більш сприятливою.

Існує кілька підходів визначення успіху проекту. Традиційний підхід полягає у використанні трикутника управління проектом, де час і вартість кожного з них утворюють лінію в трикутнику. Часто якість включається як окремий елемент (залізний трикутник).

Обсяг – це набір функціональних елементів (функцій), що надаються протягом життя проекту. Час – це фактичний час, необхідний для завершення проекту з його обсягом. Вартість – це кількість ресурсів, необхідних для завершення проекту. Внесення змін до одного з елементів трикутника вплине на інші.

Якість дуже важлива при розробці програмних продуктів. Якість можна визначити кількома способами, але для програмних продуктів часто використовуються такі види якості:

– функціональна;

– структурна;

– технологічна.

Функціональна якість означає, наскільки добре продукт працює для передбачуваного користувача. Тобто, наскільки добре програмне забезпечення відповідає визначеним вимогам та дизайну, скільки існує

помилок, наскільки воно має хорошу продуктивність і наскільки легко його вивчати та використовувати.

Структурна якість відноситься до якості вихідного коду продукту. Це включає надійність, ремонтопридатність, можливість тестування, ефективність, безпеку та те, як це відповідає визначеним практикам кодування.

Технологічна, або якість процесу відноситься до того, як створюється система та процес навколо неї. Основними атрибутами якості процесу є дотримання термінів поставки, виконання бюджетів і повторюваний процес розробки, який надійно забезпечує якісне програмне забезпечення.

Для того, щоб проект вважався успішним, він повинен бути виконаний вчасно, в рамках бюджету та з усіма необхідними функціями та функціями.

Однак не завжди легко визначити, успішний той чи інший проект чи ні. Зацікавлені сторони, які залучені до проекту, можуть мати різні погляди на те, що є успіхом.

У той час як зовнішні зацікавлені сторони зазвичай дивляться на витрати та час, внутрішні зацікавлені сторони часто використовують обсяг і якість як найважливіші критерії для визначення успіху.

Часто якість визначають якість як «сума ефективних оцінок кожним клієнтом кожного об’єкта, важливого для клієнта, що створює задоволеність клієнта». Як важливу частину функціональної якості, вони вважають задоволення споживача найважливішою частиною якості. Це також підтверджується опитуванням [19, 20], де задоволеність клієнтів/користувачів була на третьому місці за тим, як вимірюється успіх. Своєчасна доставка та якість продукції посіли перше та друге місце відповідно. Цінність бізнесу та обсяг продукції посіли четверте та п’яте місце.

Фактор успіху визначається як то , що «має йти добре , щоб забезпечити успіх». У цьому контексті успіх пов’язаний з результатом програмного проекту.

Фактори можна об’єднати в групи, які називаються практиками [21]. Комунікація, інженерні методи, процес управління проектом, люди та команда, залучення та задоволеність клієнтів, а також час прийняття рішення – це приклади ідентифікованих факторів успіху, на які, ймовірно, вплинуть методи, які використовуються в проекті.

1.3. Дослідження гнучких методів розробки програмного проекту

Різні методи розробки мають різну практику. При виборі методу для даного проекту слід враховувати ці методи, щоб переконатися, що вони відповідають проекту. Не всі методи описують багато практик, а скоріше зосереджуються на інших аспектах.

Scrum описує практики у формі подій, ролей та артефактів. Екстремальне програмування описує 24 практики у формі інженерних практик [22].

Канбан описує п'ять принципів які також можна розглядати як практику [23]. Метод можна розглядати як сукупність найкращих практик, цінностей та/або принципів, які, як було доведено, працюють для певних типів проектів. У цьому розділі я детальніше розгляну деякі з часто використовуваних методів agile.

Scrum — це ітераційна та інкрементна структура, спеціально створена для управління проектами що орієнтовані на розробку програмного забезпечення у гнучкий спосіб. Scrum був визначений приблизно в 1995 році, але отримав популярність після 2001 року, коли був заснований Agile Alliance.

Структура Scrum визначає лише три ролі:

– власник продукту;

– Scrum-майстер;

– член команди.

Власник продукту контролює відставання випуску продукту та визначає вимоги. Scrum Master організує процес Scrum. Членами команди є інші учасники. Тобто розробники, тестувальники, дизайнери тощо.

Scrum має чотири події:

– планування спринту;

– щоденний мітінг;

– огляд спринту;

– ретроспективу спринту.

Кожна подія, яка обмежена часом, призначена для швидкого прогресу та постійного потоку інформації між усіма членами команди та зовнішніми зацікавленими сторонами.

Спринт (зазвичай) — це 2-4 тижневий цикл, коли команда працює над узгодженим набором завдань.

Найважливішими артефактами є [24]:

– відставання продукту;

– відставання спринту;

– збільшення продукту.

Беклог продукту містить вимоги, які будуть розроблені в майбутньому спринті. Відставання від спринту містить вимоги, які зараз опрацьовуються в рамках поточного спринту. Приріст добутку є результатом роботи, виконаної під час спринту.

У кожному спринті, під час планування спринту власник продукту і команда вирішують, яка підмножина елементів з накопичених задач має бути включена в спринт.

Команда працює під час спринту, щоб створити потенційно відвантажуваний приріст продукту. Щодня вони проводять щоденну зустріч, щоб поділитися інформацією та вирішити потенційні проблеми.

Канбан - це більше мислення, ніж методологія. Це означає, що він не пропонує багато конкретних правил щодо того, як слідувати процесу, а радше зосереджується на правильному ставленні.

Канбан має кілька основних принципів бережливої поведінки в проекті:

– візуалізуйте робочий процес;

– обмежте незавершені роботи;

– керуйте потоком;

– зробіть процес явним;

– вдосконалюйте спільну роботу.

Кожен елемент, обраний для розробки, знаходиться в пріоритетному резерві, який називається дошкою Канбан. Дошка Канбан має кілька стовпців, кожен з яких представляє статус.

Кожен елемент знаходиться в одному зі стовпців і перетікає зліва направо під час роботи над ним. Обмеження встановлюється для кожного стовпця, вказуючи, скільки елементів колонка може містити в будь-який момент часу. Якщо забагато елементів об'єднано в одну колонку, вся команда розробників відповідає за вирішення проблем і відновлення роботи потоку.

Екстремальне програмування (XP) — це програмно орієнтований метод швидкої роботи. Кент Бек, один із перших підписників Agile Manifesto, створив XP у 1999 році [25]. Сьогодні XP визначає п'ять основних цінностей, 15 принципів, 13 основних практик і 11 супутніх практик. Вони призначені для покращення якості коду та цінності продукту, а також створення належної практики розробки. XP підтримує короткі цикли розробки (ітерації) і справді зосереджується на тому, щоб запровадити якомога більше практик.

Існує кілька варіацій Scrum, або скрам-гібридів. Два з найпопулярніших – Scrumban і Scrum/XP.

Scrumban — це модель, заснована на Scrum і Kanban. Scrumban поєднує в собі ролі, події та артефакти Scrum і дошки Kanban з обмеженням незавершеної роботи. Базова відмінність полягає в тому, що Scrumban не використовує тайм-бокс від Scrum. Замість спринту робочий процес безперервний.

Scrum/XP – це комбінація практик і правил Scrum і XP. Цей метод забезпечує середовище, де клієнт може розробити програмний продукт у той,

що найкраще відповідає вимогам бізнесу. Використовуються п'ять спільних практик: ітерації, прирости, поява, самоорганізація та співпраця.

Існує багато доступних Agile інженерні практик, що використовуються при гнучкій розробці та в традиційних проектах. Ці практики включають:

– діаграму згорання;

– використанні щоденного мітингу;

– визначення готовності;

– інтерактивна розробка;

– інструментальний дизайн;

– визначення готовності;

– використання беклогу.

Burndown діаграма (або діаграма згорання) являє собою графік , який показує , скільки роботи залишаючись в ітерації (або у всьому проекті) [26]. Він також показує, скільки часу пройшло від початку ітерації/проекту. Ця практика дає всім учасникам сучасне уявлення про стан ітерації/проекту. Ця прозорість також має на меті заохотити команду.

Щоденний мітинг – це подія, на якій команда розробників збирається щодня протягом кількох хвилин, щоб координувати розробку та поділитися важливою інформацією. Кожен учасник описує завершену роботу та будь-які перешкоди, які їм необхідно усунути, перш ніж продовжити. Зустріч має бути короткою і по суті.

Визначення готовності – це практика, коли команда приймає рішення відносно набору вимог, яким має бути задоволено, щоб елемент/історію вважався завершеним. Усі в команді повинні мати узгоджене рішення того, що означає, коли елемент/історію вважається виконаним.

Ітеративна розробка – це практика включення планування в кожен цикл розробки. Тобто практика вирішувати, який обсяг робіт для наступного циклу. Однією з переваг цієї практик є те, що ви не витрачаєте час на створення невідповідного продукту. У міру створення нових функцій відгуки від клієнтів

допомагають спрямувати продукт у правильному напрямку, а обсяг визначається та перевизначається протягом усього проекту.

Суть інкрементного проектування полягає в тому, що замість того, щоб розробляти всю систему на початку проекту, коли ви не можете точно знати, як повинен виглядати продукт, ви проектуєте лише те, що вам потрібно, з наявною інформацією. В подальшому вимоги до проекту доповнюються по мірі реалізації його функцій.

Беклог продукту – це список функцій і технічних робіт, які, ймовірно, знадобляться в поточному проекті. Його слід розставити за пріоритетами, поставивши першочергові елементи. Це повний спектр функцій, які ще не розроблені, але вони можуть змінюватися з часом.

Власник продукту грає певну роль. Бажано, щоб замовник був окремою особою, яка відповідала б за управління відставанням і переконання, що елементи, відібрані для розробки, максимізують цінність продукту.

Скрам-майстер - це слуга-лідер, основним обов'язком якого є переконатися, що команда дотримується теорії, практики та правил Scrum [23]. Скрам-майстер повинен допомагати команді максимізувати свою продуктивність у будь-який спосіб.

Планування спринту (або планування ітерації) — це подія Scrum, де команда збирається, щоб обговорити та вирішити, над чим вони працюватимуть на наступній ітерації. Скрам-майстер представляє мету майбутньої ітерації та пояснює, які елементи із відставання слід включити, щоб досягти мети спринту. Команда розробників вибирає предмети, які вони можуть зробити, і формує разом із скрам-майстером мету або окремого спринту або ітерації.

Беклог спринту — це артефакт, визначений Scrum, який містить список елементів, які команда розробників вважає, що вони можуть виконати в поточному спринті (ітерації). Визначається під час спринту планування, але може бути змінений під час спринту командою розробників. Кожен пункт спринту повинен бути добре описаний і добре зрозумілий командою.

Ретроспектива спринту – це процедура, визначена Scrum, де команда збирається після спринту, щоб обговорити, що спрацювало добре та як можна покращити. Під час цієї процедури вони оглядають останній спринт (або декілька останніх спринтів) і шукають шляхи покращення наступного спринту. Мета – зробити команду кращою та покращити процес розробки IT проекту.

Огляд спринта — це ще одна подія Scrum, де вибрані зацікавлені сторони зустрічаються з командою розробників, щоб перевірити зміни та надіслати відгук. Це не статусна зустріч, а радше неформальна зустріч, основна мета якої – одержати зворотній зв'язок, сприяти співпраці та обговорити майбутню роботу.

Практика членів команди – це практика Scrum, яка стверджує, що єдиною дозволеною роллю в Scrum-команді, за виключенням власника продукту та керівника групи розробників, є члени команди. Розробники, тестувальники, графічні дизайнери тощо вважаються членами команди і не мають іншої ролі в команді. Ця модель розроблена для розвитку гнучкості, креативності та продуктивності процесу розробки.

Історія, яка часто називається розповіддю користувача, є користуч-орієнтованим способом опису нової функції. Розповідь написана в певному стилі, щоб виділити, кому потрібна функція, що їм потрібно і навіщо вона потрібна.

Дошка завдань являє собою дошку з колонками стану, в якому кожен елемент / історія поміщається на підставі його статусу. Це звичайний спосіб показати прогрес команди, а також статус кожного елемента. Дошки завдань можуть бути цифровими або фізичними.

На дошці кабан із фізичним завданням розповідь часто просто пишеться на листівці й переміщується від колонки до колонки в міру переходу від початку до завершення .

Дошка завдань називається дошкою Канбан, і вона має властивість незавершеної роботи для кожного стовпця. Обмеження незавершеного

виробництва у дошці Канбан призначені для запобігання блокування процесу розробки та прискорення проходження елементів через дошку.

Ідея полягає в тому, що якщо стовпець містить більше елементів, ніж ліміт часу, вся команда зобов'язана допомогти та переконатися, що блок вирішено. Це обмеження – це те, що відрізняє звичайну дошку завдань від дошки Kanban.

Спільне покращення – це практика, яка заохочує членів команди постійно вдосконалювати команду та її процес за допомогою невеликих змін. Мета цієї практики дещо схожа на цілі ретроспективної практики спринту , але більше уваги приділяється покращенню процесу співпраці та способу спільної роботи людей.

Командна оцінка – це практика, коли всі члени команди, принаймні ті, хто бере участь у кодуванні, беруть участь в оцінці завдання. Це займає більше часу, коли задіяна вся команда, але в результаті виходить більш точна оцінка завдання.

Agile ігри – це загальна назва для всіх ігрових практик, які покликані зробити частину процесу в agile-командах легшою, веселішою або кращою. Одним із добре відомих прикладів гнучких ігор є планування покеру, яке є способом більш точно оцінювати історії користувачів.

Часто кажуть, що Agile-команди самоорганізуються та мають багатофункціональний характер. Самоорганізація означає, що команда сама несе відповідальність за вирішення проблем проекту, виходячи з кордонів і обмежень, встановлених керівництвом.

Міжфункціональність, яку також називають цільною командою, означає, що команда повинна бути компетентною - мати відповідні знання та навички для створення функції, яку вони розробляють. Створення групи експертів також може вплинути на вартість проекту, оскільки експерти, як правило, дорожчі ніж новачки. Однак це також зменшить кількість часу, необхідного для завершення проекту.

1.4 Аналіз систем та підходів підтримки прийняття рішень

Системи підтримки прийняття рішень (СППР) — це інформаційні системи, які сприяють процесам прийняття рішень з управління підприємством [27]. Інформаційна система підтримки рішень допомагає керувати організацією середнього і високого рівня, аналізуючи величезні обсяги неструктурованих даних і накопичуючи інформацію, яка може допомогти у вирішенні проблем щодо прийняття рішень. СППР створює детальні інформаційні звіти шляхом збору та аналізу даних.

В організації СППР використовується відділами планування, такими як операційний відділ, який збирає дані та створює звіт, який може використовуватися менеджерами для прийняття рішень. В основному, СППР використовується для прогнозування послідовності виконання робіт, для даних, зі сфери упраління запасами та операціями, а також для представлення інформації клієнтам у легкій для розуміння формі.

Системи підтримки прийняття рішень застосовуються в багатьох сферах знань, зокрема управління ІТ проектами, управлінні медичними послугами та ін. Одним з основних застосувань СППР в організації є звітування в режимі реального часу. Це може бути дуже корисним для організацій, які беруть участь в управлінні за принципом « точно вчасно» (JIT) .

Підхід JIT потребує даних про рівень ресурсів (в тому числі людських ресурсів) у режимі реального часу, щоб розмістити замовлення «точно вчасно», щоб запобігти затримкам у виробництві та викликати негативний ефект доміно. Реалізація такого підходу у процесі розробки ІТ проекту стосується управління пріоритетами задач, які реалізуються в проекті. Зміна пріоритетів приводить до зміни послідовності робіт проекту. Це дає можливість врахувати строки завершення окремих завдань.

Система підтримки прийняття рішень містить з трьох основних складників: модуль підтримки рішень на основі моделі; модуль підтримки рішень на основі знань; інтерфейс з користувачем (рис. 1.4) [28].

Рисунок 1.4 – Підтримка рішень на основі моделі та на основі знань в СППР

Модуль підтримки рішень на основі моделі зберігає моделі, які менеджери можуть використовувати при прийнятті рішень. Моделі використовуються у процесі прийняття рішень щодо фінансового стану організації, прогнозуванні попиту на товар чи послугу, а також можуть бути використані при плануванні робіт над проектом.

Модуль підтримки рішень на основі знань використовує інформацію з внутрішніх джерел (інформація, зібрана в системі транзакцій) і зовнішніх джерел (онлайн-бази даних). На основі цієї інформації формуються знання щодо об'єкту управління.

Відмінність між модулями полягає в тому, що в першому модулі використовується інформація щодо процесів на об'єкті управління, а в другому моделі – безпосередньо знання про процеси управління.

Інтерфейс користувача включає інструменти, які допомагають кінцевому користувачеві СППР використовувати системи підтримки прийняття рішень. В системах підтримки прийняття рішень використовується як графічний, так і текстовий інтерфейс.

Системи підтримки прийняття рішень мають такі переваги, пов'язані із удосконаленням процесу управління:

– підвищується ефективність діяльності з прийняття рішень, оскільки СППР може збирати та аналізувати дані в реальному часі;

– сприяє навчанню в організації, оскільки для впровадження та запуску СППР в організації необхідно розвивати конкретні навички;

– автоматизуються монотонні управлінські процеси, що означає, що керівник може більше часу витрачати на прийняття рішень;

– покращується міжособистісне спілкування в організації.

Слід також зазначити, що СППР мають наступні недоліки:

– вартість розробки та впровадження СППР є високою, що робить такі системи менш доступним для невеликих організацій;

– компанія може розвинути залежність від СППР, коли вона інтегрована в процеси прийняття рішень протягом кожного дня для підвищення ефективності та швидкості; однак керівники схильні занадто покладатися на систему, що позбавляє суб’єктивного аспекту прийняття рішень;

– СППР може призвести до інформаційного перевантаження, оскільки інформаційна система схильна розглядати всі аспекти проблеми; це створює дилему для кінцевих користувачів, оскільки вони мають дуже широкий вибір альтернатив рішення;

– впровадження СППР може викликати страх і негативну реакцію з боку співробітників нижчого рівня; багатьох з них не влаштовують нові технології і вони бояться втратити роботу через ці технології.

На сьогодні використовується кілька підходів до прийняття рішень, які реалізують різні варіанти процесу, за допомогою якого керівники приходять до своїх рішень. Такі рішення можуть бути раціональними або частково раціональними.

Раціональне рішення містить всі варіанти дій, які можливі, а частково раціональне – підмножину таких варіантів. Перелік підходів та їх властивості

наведено в табл. 1.1. Перший підхід забезпечує раціональне рішення, а інші є обмежено раціональними.

Таблиця 1.1 – Підходи до підтримки прийняття рішень

Підхід	Особливості
1. Раціональний	Особи, які приймають рішення, є абсолютно об’єктивними та мають повну інформацію щодо об'єкту управління.
2. Процесний (поведінковий)	Особи, які приймають рішення, обмежують вхідні дані в процесі прийняття рішень, зосереджують свою увагу на двох-трьох найбільш сприятливих альтернативах (особливо з урахуванням обмежень в часі), обробляють ці альтернативи детально і базують свої рішення на судженнях та особистих рішеннях.
3. Гібридний	Особа, яка приймає рішення, має вийти за рамки практичних правил і обмежень і створити якомога більше альтернатив з використанням заданих ресурсів (часу, грошей та інших практичних аспектів ситуації).
4. Персональний	Використовується модель конфлікту для прийняття рішень

Раціональний підхід до прийняття рішень реалізує системний, поетапний процес прийняття рішень. Ця концепція визначає, що організація керується особами, які приймають рішення, та які є абсолютно об’єктивними й мають повну інформацію про цю організацію [29].

Раціональний підхід має кілька сильних сторін. Той, хто приймає рішення, має розглядати рішення логічним, послідовним чином, а поглиблений аналіз альтернатив допомагає йому робити вибір на основі інформації, а не особистих упереджень, емоцій чи соціального тиску.

Однак його слабкі сторони полягають у тому, що менеджер не завжди володіє ідеальною інформацією, стикається з часовими та фінансовими обмеженнями, може мати обмежену здатність обробляти інформацію і не може точно передбачити майбутнє. Крім того, усі альтернативи не можна оцінити кількісно, що ускладнює порівняння.

Раціональний підхід є «ідеальним» підходом, що потребує повної інформації щодо предметної області. Така властивість приводить до труднощів при його практичному використанні.

Підходи 2-4 є обмежено раціональними. Переважне застосування при розробці IT проектів знайшов підхід 2 внаслідок обмежень на час виконання проектів.

Процесний, або поведінковий підхід передбачає, що особи, які приймають рішення, діють з обмеженою раціональністю, а не з ідеальною раціональністю, яку передбачає раціональний підхід. Обмежена раціональність базується на ідеї про те, що особи, які приймають рішення, не можуть мати справу з інформацією про всі аспекти та альтернативи, що стосуються проблеми, і тому вирішують розглянути деяку її значущу частину.

Таким чином, цей процес не є цілком раціональним, і тому отримані рішення не є цілком ідеальними. Особи, які приймають рішення, діючи з обмеженою раціональністю, обмежують вхідні дані в процес прийняття рішень, зосереджують свою увагу на двох-трьох найбільш сприятливих альтернативах (особливо якщо є обмеження в часі), обробляють їх дуже детально і базують свої рішення на судженнях та особистих рішеннях. упередженості, а також на логіці.

Цей підхід має такі особливості:

– використання процедур і практичних правил, які зменшують невизначеність у прийнятті рішень на початковому етапі. Наприклад, було виявлено, що використання моделей навчання покращує результативність здобувачів у минулому. Тому викладачі, знаючи такий зв’язок, вирішують використовувати моделі навчання;

– підоптимізація, яка відноситься до свідомого прийняття меншого, ніж найкращий можливий результат, щоб обминути ненавмисний негативний вплив на інші аспекти організації;

– альтернативи розглядаються лише до тих пір, поки не буде знайдено рішення, яке відповідає мінімальним вимогам; а потім додаткові зусилля не докладаються. Пошук альтернатив, як правило, є послідовним процесом, заснованим на процедурах і практичних правилах, які керуються попереднім досвідом з подібними проблемами.

Отримане рішення не завжди може бути оптимальним, оскільки пошук часто завершується, коли визначається перша мінімально прийнятна альтернатива.

Гібридний підхід поєднує кроки раціонального підходу з особливостями та умовами поведінкового підходу для створення більш реалістичного процесу прийняття рішень.

Відповідно до цього підходу, замість того, щоб генерувати всі альтернативи, особа, яка приймає рішення, повинна намагатися вийти за рамки практичних правил і обмежень і створити якомога більше альтернатив протягом заданого часу, грошей та інших практичних аспектів ситуації. Тут раціональний підхід забезпечує аналітичну основу для прийняття рішень, тоді як поведінковий підхід забезпечує модеруючий вплив.

Попередні три підходи чітко пояснюють процеси, які беруть участь у прийнятті рішень. Однак вони не визначають те, як люди приймають рішення, коли вони нервують, занепокоєні, стурбовані чи схвильовані – чи то в організації, чи в особистих справах.

Більш реалістичний погляд на індивідуальне прийняття рішень базується на моделі конфлікту. Ця модель заснована на дослідженнях соціальної психології та індивідуальних процесів прийняття рішень. Це особистий підхід до прийняття рішень, оскільки він має справу з особистими конфліктами, з якими люди стикаються в особливо складних ситуаціях прийняття рішень.

Модель конфлікту має п'ять основних характеристик:

– Модель стосується лише важливих життєвих рішень, таких як вибір характеру та типу освіти та навчального закладу, кар’єри, шлюбу, основних організаційних рішень тощо, які зобов’язують особу чи установу до певного курсу дій;

– Модель визнає, що прокрастинація та раціоналізація є механізмами, за допомогою яких люди уникають приймати важкі рішення та справлятися з супутнім стресом.

Модель визнає, що деякі рішення можуть піти не так.

– Конфлікт передбачає самореакцію в термінах порівняння альтернатив з внутрішніми моральними стандартами. Якщо певний спосіб дій порушує моральні переконання осіб, які приймають рішення, навряд чи буде обраний, навіть якщо він є економічно та соціально вигідним.

– Конфлікт визнає, що іноді особа, яка приймає рішення, неоднозначно ставиться до альтернативних напрямків дій. Така ситуація заважає йому віддатися єдиному рішенню від усього серця. Однак важливі рішення, що стосуються власного життя, - це рішення або-або, які вимагають прихильності до однієї конкретної альтернативи, не допускаючи значних компромісів.

Відповідно до конфліктної моделі прийняття рішень, людина, стикаючись з проблемою, аналізує ситуацію, шукаючи зворотний зв’язок (можливо негативний) і запитує себе, чи серйозні ризики, пов’язані з цим, якщо вона не зробить змінити.

Якщо відповідь негативна, особа продовжить свою поточну діяльність. Тобто ця ситуація тягне за собою продовження поточної діяльності, якщо це не несе серйозних ризиків. З іншого боку, якщо ризики серйозні, то особа, що приймає рішення, вживає необхідних дій, щоб здійснити бажані зміни. Ця ситуація відома як нескладна зміна, яка передбачає внесення змін у поточну діяльність, якщо це не представляє серйозних ризиків.

Крім того, модель конфлікту також пояснює концепцію оборонного уникнення, що тягне за собою не внесення змін у поточну діяльність та

уникнення будь-яких подальших контактів із супутніми проблемами, оскільки, схоже, немає надії знайти краще рішення.

Якщо людина має мало часу на обмірковування, чи потрібно їй змінитися, можливо, через свій похилий вік, вона відчує підвищену пильність, через що може зазнати сильного психологічного стресу та братися за шалене, поверхневе пошук якоїсь стратегії, що приносить задоволення.

З іншого боку, якщо у людини є два-три роки, щоб розглянути різні альтернативи, вона буде проводити детальну обробку інформації, що означає, що вона буде ретельно досліджувати всі можливі альтернативи, зважувати їх витрати та вигоди, перш ніж приймати рішення та розробляти плани на випадок непередбачених обставин.

1.4 Постановка задачі дослідження

Актуальність дослідження полягає в такому. Існуючі методи підтримки рішень при розробці IT проектів орієнтовані на гнучку технологію розробки та забезпечують уточнення послідовності робіт на основі шляхом пріоритетів для задач IT проекту. Пріоритети виставляються на основі оцінок користувачів, оцінок власника проекту. Однак необхідність постійного удосконалення процесу розробки при зміні оцінок користувачів, затримках у розробці задач потребують уточнення пріоритетів на основі інтеграції даних підходів.

Об’єктом дослідження кваліфікаційної роботи є процес розробки IT проекту.

Предметом дослідження є методи підтримки прийняття рішень для удосконалення процесу розробки IT проекту з використанням даних щодо послідовності дій такого процесу.

Метою даної роботи є дослідження методів автоматизованої підтримки прийняття рішень для підвищення ефективності процесу розробки ІТ проекту на основі аналізу послідовності дій такого процесу.

Для досягнення мети магістерської роботи необхідно вирішити такі задачі дослідження:

– аналіз процесу розробки ІТ проекту;

– структуризація процесу підтримки прийняття рішень щодо удосконалення процесу розробки ІТ проекту;

– удосконалення методу визначення пріоритетів розробки задач ІТ проекту з використанням моделі задоволення потреб Кано;

– планування проекту розробки методу визначення пріоритетів розробки задач ІТ проекту;

– експериментальна перевірка отриманого удосконаленого методу визначення пріоритетів розробки задач ІТ проекту.

2. ПІДТРИМКА ПРИЙНЯТТЯ РІШЕНЬ У ПРОЦЕСІ РОЗРОБКИ ІТ ПРОЕКТУ

2.1 Підтримка прийняття рішень щодо удосконалення процесу розробки ІТ проекту

Гнучкі методи продовжують залишатися стратегією вибору для більшості компаній-розробників програмного забезпечення та все частіше для організацій, які не займаються технологіями. Згідно зі звітом Інституту управління проектами, 71% організацій повідомляють про використання гнучких підходів. Частково це пов’язано з тим, що agile проекти на 28% успішніші за традиційні проекти, за даними PwC.

Підтримка прийняття рішень в межах гнучкого процесу розробки ІТ проекту містить у собі етапи, представлені на рис. 2.1.

На першому етапі визначається мета прийняття рішення для конкретної ретроспективи. Деякі підходи до підтримки прийняття рішень не передбачають формулювання мети. Однак цей етап для процесу розробки ІТ проекту є доцільним, оскільки його можна використовувати як стандарт для визначення відповідності прийнятого пізніше рішення.

На другому етапі розпізнається та діагностується проблема, яка потребує вирішення. Передумовою даного етапу є розуміння природи, масштабу та причин проблеми.

На даному етапі виконується збір інформації, яка має відношення до мети. Якщо існує невідповідність між фактичною ситуацією та метою, можуть знадобитися додаткові дії. Достовірна інформація тут є абсолютно необхідною. Неточна інформація може призвести до неправильних рішень. На цьому етапі також визначаються обмеження, в межах яких проблема має бути вирішена.

На третьому етапі особи, які приймають рішення, повинні визначити, чи вимагає проблема запрограмованого чи незапрограмованого рішення. Якщо

потрібне запрограмоване рішення, то робиться вибір із уже відомих доступних альтернатив. Якщо ж рішення є новим, невідомим, то потрібно створити нові альтернативи.

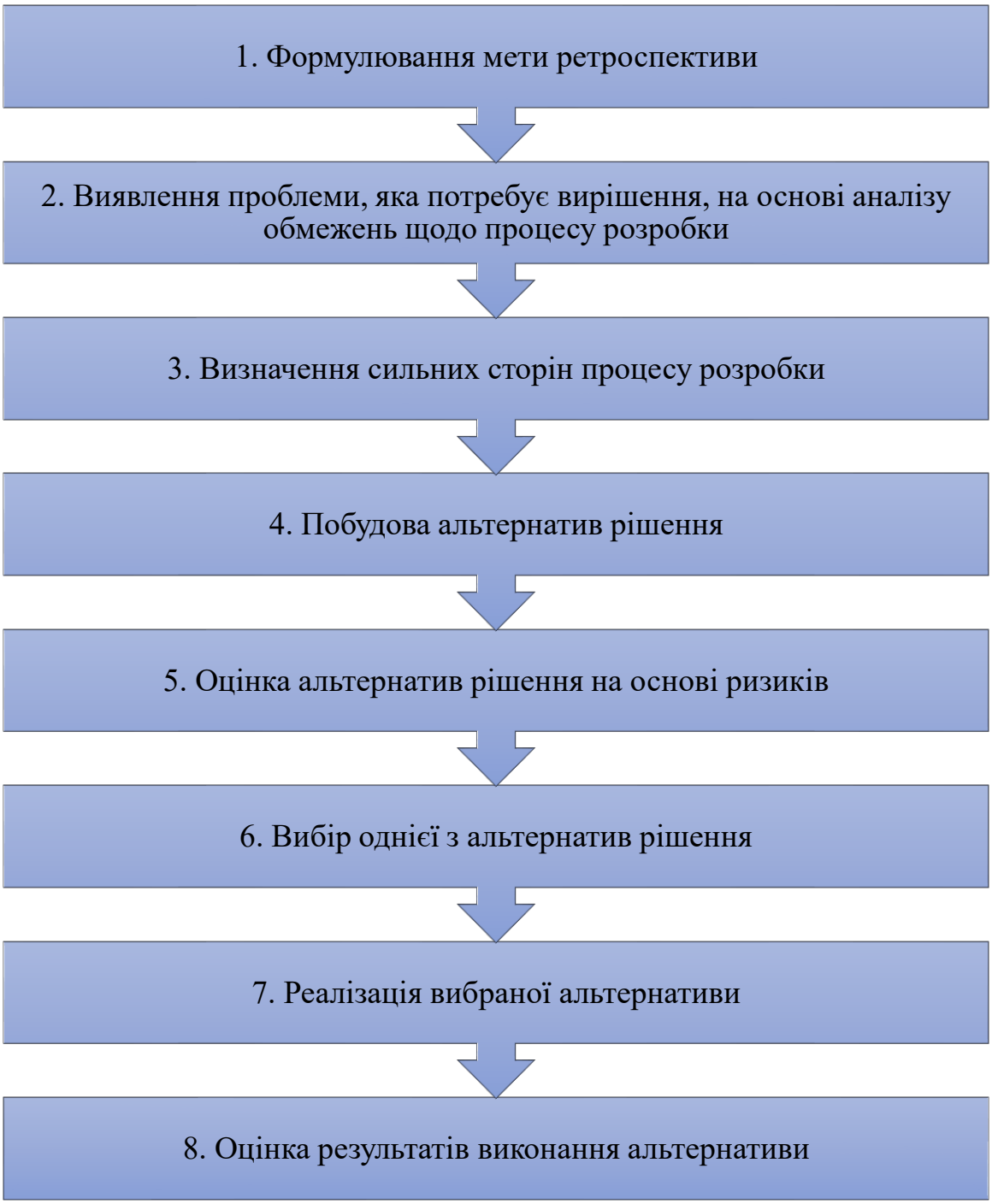

Рисунок 2.1 – Послідовність етапів раціонального підходу до підтримки прийняття рішень

Четвертим етапом у прийнятті незапрограмованого рішення є створення альтернатив. Тут особи, які приймають рішення, генерують альтернативи на основі своєї освітньої підготовки, а також професійного досвіду та знань про ситуацію. Крім того, інформацію можна отримати від колег, підлеглих, експертів та керівників.

Особи, які приймають рішення, можуть аналізувати симптоми проблем, щоб знайти підказки, або покладатися на власну інтуїцію чи судження, щоб знайти альтернативні рішення.

На п'ятому етапі кожна альтернатива оцінюється з точки зору її сильних і слабких сторін, витрат і вигід, а також ризиків, які стосуються можливих негативних наслідків відповідно до попередньо визначених критеріїв щодо прийняття рішень. Позитивні наслідки необхідно зважувати з негативними наслідками.

Основним критерієм рішення тут є те, чи наблизить нас конкретне рішення до мети. Процес оцінки нових альтернатив зазвичай включає такі кроки:

– опис очікуваних результатів кожної альтернативи;

– оцінку очікуваних витрат кожної альтернативи;

– оцінку невизначеності та ризиків, пов'язаних з кожною альтернативою рішення.

У більшості ситуацій особи, які приймають рішення, не мають повної інформації щодо результатів усіх альтернатив, які є у їхньому розпорядженні.

Вибір альтернативи є найважливішим етапом у процесі прийняття рішення. На даному етапі вибирається найкраща альтернатива з максимальними позитивними наслідками, найменшими негативними наслідками або без них, меншими ризиками та мінімальними витратами. Іншими словами, визначається очікуване значення кожної альтернативи і вибирається альтернатива з найбільшими очікуваними значеннями. Вибір альтернативи залежить від освіти, суджень, досвіду, логічного аналізу та інших показників.

На шостому етапі здійснюється вибір однієї з альтернатив, що дозволить розв'язати поточну проблему, як виявили на етапі 2. Такі альтернативні дії слід використати, якщо основна послідовність дій несподівано порушена або стає невідповідною стану підприємства. Планування послідовності дій у непередбаченій ситуації є частиною переходу між вибором бажаної альтернативи та її реалізацією.

На сьомому етапі після офіційного прийняття рішення щодо управління підприємством видається дозвіл на його виконання. При впровадженні рішення реалізується, включаючи повідомлення про рішення, придбання та призначення ресурсів для забезпечення його виконання. Реалізація вибраного варіанту рішення ґрунтується на мотивації тих, хто приймає участь у процесі прийняття рішень.

Успішне впровадження вимагає належного використання ресурсів і хороших управлінських навичок, лідерських якостей, структури винагород, знань і застосування групової динаміки. Іноді той, хто приймає рішення, починає сумніватися в уже зробленому виборі. Ця ситуація розглядається як проблема когнітивного дисонансу. Когнітивний дисонанс - це занепокоєння, яке відчуває людина, коли два набори знань або сприйняття несумісні або суперечать один одному. Щоб зменшити когнітивний дисонанс, особа, яка приймає рішення, може шукати додаткового раціоналізації рішення за допомогою нової інформації.

Це останній етап процесу прийняття рішень – етап контролю, на якому результати рішення вимірюються в порівнянні з попередньо визначеними, бажаними цілями. Якщо між ними є розбіжність, особа, яка приймає рішення, може перезапустити процес прийняття рішень шляхом перегляду/модифікації/встановлення нових цілей.

Удосконалення процесу в межах гнучкої методології розробки виконується на етапах 2 та 4. На другому етапі виявляються проблеми, обмеження, а також сильні сторони процесу розробки та підході до вирішення

проблем. На четвертому етапі формуються рішення, що задають послідовність робіт IT проекту.

Ретроспективи відіграють важливу роль як віхи після спринту для будь-якого масштабного проекту. Знання того, як отримати максимальну віддачу від кожної зустрічі, допомагає визначити моменти для покращення, отримати більше відгуків від ваших команд і визначити свій наступний спринт для успіху.

Ефективні ретроспективи є життєво важливими для гнучкого процесу розробки IT проекту. Вони можуть привернути увагу до корисних змін, які слід внести, щоб допомогти визначити покращення процесу для наступної ретроспективної зустрічі та підштовхнути команду до більш ідеального процесу для майбутніх спринтів на шляху до завершення IT проекту.

У кращому випадку, невдало проведена ретроспективна зустріч спринту є рутинною і нудною. Без активної участі команда не може відфільтрувати шум, щоб визначити реальні покращення процесу, які можуть мати глибокий вплив. У гіршому випадку, погано проведені ретроспективні зустрічі можуть шкодити результатам проекту

У звіті Quantum’s Workplace 2015 Employee Engagement Report головні чинники залучення включали довіру до лідерів і віру в майбутній успіх організації. Ефективні ретроспективи дають можливість команді час і платформу, щоб розглянути деталі, які кожен з них вважає доречними, а також відповісти на запитання, які у них можуть виникнути про попередній етап.

Створення прозорості в ретроспективі допомагає задати тон проекту і може покращити взаємодію співробітників у вашій команді.

У практиці agile власник продукту не повинен бути єдиною особою, яка має право голосу щодо зміни процесу розробки. Корисна ретроспектива після спринту може змінити результати цілих проектів. Ретроспектива є інструментом для зменшення ризику. Вони дозволяють виявляти проблеми та виправляти їх якомога раніше.

При ранньому виявленні дрібних проблем незначні, поетапні зміни, які відбуваються в результаті усунення цих проблем, можуть згодом призвести до набагато більш ефективних результатів. Вирішення дрібних проблем з часом збільшується.

Можливо, найважливіша перевага великої ретроспективної зустрічі полягає у вдосконаленні процесу розробки IT проекту. Принципи Agile значною мірою охоплюють постійне навчання та вдосконалення, а ефективні ретроспективи служать основним інструментом у цьому процесі постійного вдосконалення. Однак на практиці лише 4% компаній вимірюють і керують своїми документованими процесами. Тому ефективні ретроспективи можуть допомогти вам покращити процеси

Під час ретроспективних зустрічей ваші співробітники можуть запропонувати рішення проблем, які були виявлені під час зустрічі, знайти можливості для покращення ваших процесів і визначити зміни в проекті, які були б корисними. З часом ці вдосконалення процесу покращать ефективність кожного спринту та призведуть до кращих сукупних результатів.

На етапі ретроспективи виконується підтримка прийняття рішень щодо удосконалення процесу розробки програмного проекту.

Але в процесі проведення ретроспектив виникає ряд суттєвих проблем, що може вплинути на удосконалення процесу розробки IT проекту:

- нечітко задана мета ретроспективи;

– не виділено ключові проблеми у розробці IT проекту;

– відсутність інструментальної підтримки ретроспективи у процесі розробки IT проекту;

– неповнота та неточність рекомендацій з удосконалення процесу розробки IT проекту;

– відсутність інформації про минулі дії зі спринту або проекту.

Для формування мети необхідно визначити закономірності у процесі розробки, вирішити, що важливо, а потім обговорити, що потрібно зробити,

щоб покращити. При виділенні ключових проблем використовуються такі підходи:

– включаються якомога більше зацікавлених сторін;

– зберігається анонімність.

Інструменти для ретроспективи мають фіксувати та впорядковувати всі відгуки про вашу команду, а також упорядковувати рекомендації від кожного члена команди в легкозасвоюваний формат.

Вибір кращих рекомендацій здійснюється шляхом голосування. Коли запропонована зміна чи дія здається занадто великими, щоб їх коли-небудь завершити, необхідно визначити найменший перший крок для удосконалення процесу розробки проекту.

При проведенні ретроспективи оцінюється успіх попереднього спринту. Виявляється, чи були вдалими ті зміни, які були обговорені та впроваджене при реалізації ІТ проекту. З’ясовується , що можна зробити краще у вашому поточному спринті, ніж у попередньому.

Для вирішення останньої проблеми ретроспектива має починатись з огляду спринту до ретроспективи, де порівнюється те, що планувалось та те, що дійсно було зроблено.

Порівняння можливостей існуючих підходів до підтримки прийнятті рішень на етапі ретроспективи наведено у табл. 2.1.

Ретроспектива Scrum на основі сильних сторін процесу розробки призначена для того, щоб визначити, де полягають сильні сторони команди, і використовувати їх, щоб стимулювати зміни та покращення під час наступного спринту. Ретроспектива включає два етапи: виявлення сильних сторін; визначення дій, які використовують ці сильні сторони.

Ретроспектива на основі перешкод у досягненні цілей послідовно використовує такі дії.

По-перше, команда враховує перешкоди (блокатори), які можуть завадити їй досягти своїх цілей.

Таблиця 2.1 – Порівняння можливостей існуючих підходів до підтримки прийнятті рішень на етапі ретроспективи

Підхід	Послідовність етапів
1. Ретроспектива Scrum на основі сильних сторін процесу розробки	– виявлення сильних сторін; – визначення дій, які використовують ці сильні сторони;
2. Ретроспектива на основі перешкод у досягненні цілей	– опис обмежень (перешкод) із досягнення цілей; – визначення ризиків; – визначення умов задоволення обмежень;
3. Ретроспектива зі зворотним зв'язком	- попарне (циклічне) обговорення проблем; - обговорення можливих рішень після циклічного обговорення проблем.

Ретроспектива зі зворотним зв'язком полягає в тому, що кожен член команди надає відгук партнеру, з яким вони стикаються. Після завершення першого раунду зворотного зв’язку учасники змінюються пари для обговорення.

Таким чином, підтримка прийняття рішень щодо удосконалення процесу розробки ІТ проекту виконується на етапі ретроспективи і полягає у виявленні «вузьких місць» й подальшому удосконаленні процесу розробки та містить вирішення таких ключових задач:

– виявлення проблеми, що потребує вирішення на основі перешкод із досягнення цілей;

– побудова й оцінка альтернатив рішення на основі виявлення сильних сторін процесу розробки, а також ризиків;

– вибір та виконання альтернативи;

– оцінка результатів виконання.

Таким чином, ретроспектива дає можливість виявити проблеми процесу розробки. Вирішення цих проблем задається на етапі побудови альтернатив

рішення з удосконалення процесу розробки ІТ проекту. Кожна з альтернатив задає послідовність дій по реалізації проекту. Така послідовність визначається пріоритетами задач проекту.

Тому підтримка визначення пріоритетів дає можливість удосконалити процес управління ІТ проектом.

2.2 Удосконалення методу визначення пріоритетів задач у процесі розробки ІТ проекту на основі моделі задоволення потреб Кано

Виставлення пріоритетів задач, що входять у беклог, в ІТ проектах виконується за допомогою таких методів:

– stack ranking;

– метод на основі моделі задоволення потреб Кано;

– метод MoSCoW;

– визначення пріоритетів на основі вартості затримки.

Характеристики даних методів наведено у табл. 2.2 [24].

Таблиця 2.2 – Методи визначення пріоритетів задач

Підхід	Ключові характеристики
Визначення пріорітетів за часом відставання (Stack Ranking)	Пріоритет задачі визначається за відставання від заданих строків завершення
Визначення пріоритетів на основі моделі задоволення потреб Кано	Задачі класифікуються відповідно до потреб і очікувань клієнтів за такими їх властивостями з точки зору користувача: обов’язкова, приваблива, одновимірна, байдужа і зворотна.

Кінець таблиці 2.2

Метод MoSCoW	Задачі упорядковуються за наступними характеристиками: Must Have (обов'язкові); Must Have (потрібні, але не самі важливі); Could Have (бажані); Would (можуть бути перенесені на наступні релізи).
Визначення пріоритетів на основі вартості затримки	Використовується лінійний розрахунок вартості по часу затримки, обмеження по даті. Враховується зміна вартості затримки в залежності від дати.

Наведені методи орієнтовані на agile технологію розробки та забезпечують коригування послідовності робіт шляхом зміни пріоритетів для задач ІТ проекту, які встановлюються з урахуванням оцінок користувачів та власника проекту. Однак необхідність постійного удосконалення процесу розробки при зміні оцінок користувачів, затримках у розробці задач потребують уточнення пріоритетів на основі інтеграції даних підходів. Тому в роботі удосконалено метод визначення пріоритетів задач ІТ проекту на основі моделі задоволення потреб Кано

Узагальнене представлення удосконаленого методу наведено на рис. 2.2.

Метод використовує як вхідні дані інформацію в балах щодо важливості j – задачі для користувачів за градаціями:

– $a_{1,j}$ «очікую»;

– $a_{2,j}$ «все рівно»;

– $a_{3,j}$ «не подобається». Інформація надається для наявності $a_{i,j}^{+}$ та відсутності $a_{i,j}^{-}$ відповідної функції.

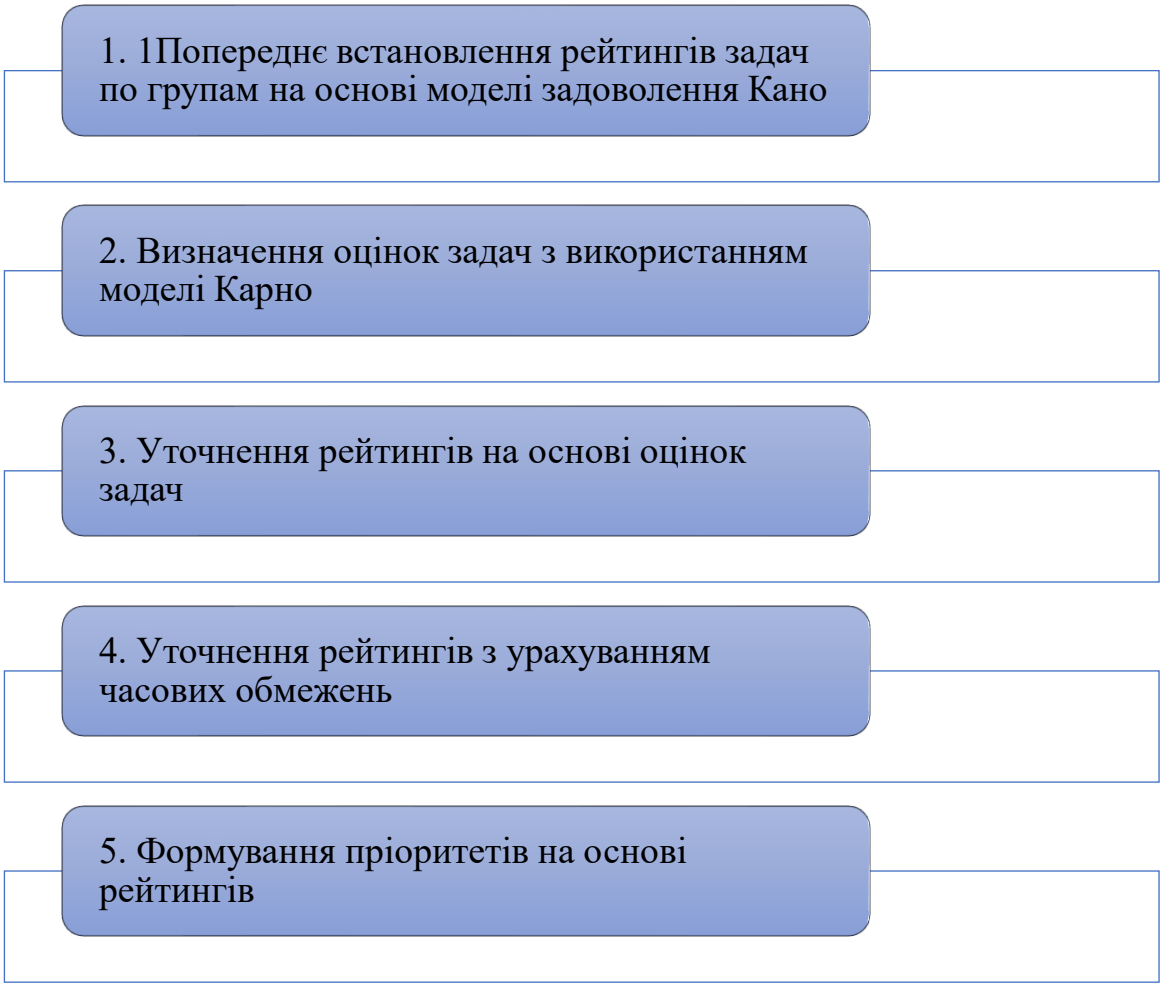

Рисунок 2.2 – Узагальнене представлення удосконаленого методу визначення пріоритетів задач ІТ проекту на основі моделі задоволення потреб Кано

Метод містить у собі такі етапи:

Етап 1. Встановлення рейтингів на основі моделі задоволення Кано

Крок 1.1. Розрахунок суми балів $g_{i,j}^{+}$ та $g_{i,j}^{-}$ для кожної градації $a_{i,j}^{+}$ та $a_{i,j}^{-}$ відповідно.

Крок.1.2 Побудова характеристик задачі за категоріями на основі суми балів $g_{i,j}^{+}$ та $g_{i,j}^{-}$ за табличною моделлю Карно. Згідно моделі використовуються такі градації: A – привабливі для користувача; O – лінійні; M – обов'язкові; R –зворотні (непотрібні функції).

Крок 1.3. Встановлення рейтингів для груп M, A, O, R з максимальним пріоритетом у групі M та нульовим пріоритетом у групі R.

Результатом етапу є множина попередніх рейтингів задач $P^{(1)} = \left\{ p_j^{(1)} \right\}$.

Етап 2. Визначення нормованих оцінок кожної задачі для користувача для ситуації реалізації задачі:

$$g_j^+ = \max\left\{ g_{i,j}^+ \right\}, \tag{2.1}$$

та нереалізованої задачі:

$$g_j^- = \min\left(g_{i,j}^- \right) \Big| (\forall j) g_j^+ = \max\left\{ g_{i,j}^+ \right\}. \tag{2.2}$$

Етап 3 .Уточнення рейтингів для кожної j – задачі на основі оцінок g_j^+ та g_j^-:

$$r_j^{(2)} = r_j^1 + \left(g_j^+ - g_j^- \right). \tag{2.3}$$

Етап 4. Уточнення рейтингів для задач на основі розрахункового Δt_j та максимально допустимого $\Delta t_j^{\max}$ часу виконання задачі з урахуванням порогового значення відхилення ε:

$$r_j^{(3)} = r_j^{(2)} + 1 \; if \left(\left| \frac{\Delta t_j^{\max} - \Delta t_j}{\Delta t_j^{\max}} \right| > \varepsilon, \; \Delta t_j > \Delta t_j^{\max} \right), \tag{2.4}$$

$$p_j^{(3)} = p_j^{(2)} - 1 \; if \left(\left| \frac{\Delta t_j - \Delta t_j^{\max}}{\Delta t_j^{\max}} \right| > \varepsilon, \; \Delta t_j < \Delta t_j^{\max} \right). \tag{2.5}$$

Етап 5. Встановлення пріоритетів p_j на основі упорядкування задач за рейтингами $r_j^{(3)}$.

Метод на етапах 3 та 4 додатково враховує бали користувачів щодо задач та затримки при розробці проекту. Ці бали визначають рейтинг. Рейтинг конвертується у пріоритети на етапі 5. Метод дає можливість постійно змінювати процес розробки IT проекту шляхом зміни послідовності виконання задач.

3. ПРОЕКТ РОЗРОБКИ МЕТОДУ ВИЗНАЧЕННЯ ПРІОРИТЕТІВ РОЗРОБКИ ЗАДАЧ ІТ ПРОЕКТУ

В рамках проекту розробляється модуль уточнення пріоритетів для історій користувача в рамках процесу гнучкої розробки ІТ проектів. Додаток призначений для уточнення послідовності робіт процесу розробки на основі динамічного уточнення пріоритетів завдань (історій).

Для уточнення пріоритетів використовується інформація щодо вподобань користувачів, що використовують результати розробки ІТ проекту. Дана інформація структурується по важливості задач для користувачів у разі наявності або відсутності відповідних функцій. Тобто користувачі зазначають, чи важливої для них є реалізація функції і чи будуть вони використовувати продукт у разі відсутності цієї функції. В результаті задачі ІТ проекту розділяються на такі, без яких користувач не використовуватиме продукт, дуже важливі й цікаві для користувача, а також задачі, функціональні можливості яких лінійно впливають на перевагу продукту з точки зору користувача.

Тому при розробці проекту в першу чергу важливо запустити необхідні задачі, потім цікаві для користувача, а вже потім лінійні. Такі можливості розробленого модулю забезпечують переваги при плануванні ІТ проекту в рамках гнучкої методології розробки.

Надання релевантної інформації щодо рейтингів задач (історій користувача) дає можливість з мінімальними зусиллями удосконалювати процес розробки ІТ проекту.

Обмеження проекту пов'язані із релевантністю вхідної інформації від користувачів. Так, при використанні методу МосКва важливість задач задається власником проекту та розробниками. Така інформація є релевантною, але може не враховувати останні зміни на ринку. При опитуванні користувачів додатково має сенс враховувати їх кваліфікацію.

3.1 Опис проекту для методу визначення пріоритетів

Команда проекту з розробки методу визначення пріоритетів складається із таких кваліфікованих спеціалістів:

– менеджер проектів;

– розробник (devloper);

– тестувальник.

Оцінка тривалості проекту складає 32 дні. Для розробки використовується гнучка (agile) модель.

Даний проект складається із таких задач:

– проектування із визначенням та уточненням вимог; такі вимоги формуються у вигляді user story. Останні представляють собою узагальнений опис функціональності задачі з точки зору користувача;

– розробка алгоритму уточнення пріоритетів з використанням моделі користувацьких переваг, запропонованої японським вченим Карно;

– модульне тестування з метою перевірки розробленого алгоритму уточнення пріоритетів;

– розробка візуального представлення змінених пріоритетів задач IT проекту;

– інтеграція розробленого модулю із системою підтримки розробки за технологією Scrum;

– інтеграційне тестування розробленого додатку;

– тестування інтерфейсу з користувачем.

3.2 Статут проекту

Статут призначений для формальної авторизації проекту та випускається зовнішнім керівником. Це дає можливість використовувати

організаційні ресурси в ситуації, коли керівництво несе відповідальність за проект, але не має влади над командою проекту.

Причини розробки проекту пов'язані із важливістю своєчасно враховувати зміни на ринку при розробці задач IT проекту. Такі зміни можна виявити шляхом порівняльного аналізу результатів опитування користувачів.

Мета проекту полягає в тому, щоб створити модуль уточнення пріоритетів, який може бути інтегрований до існуючих систем підтримки процесу розробки за методологією Scrum/

Результати та очікувані вигоди від виконання проекту пов'язані із поліпшенням процесу розробки та скороченням часу виводу на ринок актуальних для користувачів функцій, що збільшує кількість клієнтів фірми-розробника.

Обмеження та допущення даного проекту. Строки проекту з 8 листопада по 10 грудня 2021 року.

Ризики проекту пов'язані із внесенням значних правок на останніх стадіях розробки.

Завдання проекту:

– розробка удосконалено методу визначення пріоритетів;

– розробка програмного модулю визначення пріоритетів.

Критерієм успішного завершення проекту є схвальні відгуки від користувачів системи підтримки технології гнучкої розробки IT проекту про їх задоволеність можливостями встановлення пріоритетів.

Межі проекту визначення пріоритетів залежать від переліку робіт включаючи планування та передачу модулю замовники.

3.3 Планування проекту

Процеси планування проекту мають такі властивості: виконуються на протязі життєвого циклу проекту; починаються з побудови попереднього

повного плану, що входить до складу концепції проекту; закінчуються детальним плануванням робіт на фазі завершення проекту. Процеси планування забезпечують уточнення і деталізацію планів проекту на його життєвому циклі.

У процесі планування задаються методи і засоби управління реалізацією проекту. Проект розглядається як цілісна система. Також в процесі планування розглядаються окремі елементи проекту та етапи його виконання.

Метою планування є побудова моделі виконання проекту. За результатами планування отримуємо статут проекту. Статут об'єднує результати планування з усіх аспектів управління проектом.

Об'єктами планування в проекті є: предметна область (сукупність продукції та послуг як результату проекту), час, вартість, а також якість, ризики, тощо.

Організація спільної роботи учасників проекту виконується на основі календарних планів. Ці плани містять: строки виконання, тривалості робіт і ін. Календарний план є проектно-технологічним документом, який описує детальний перелік робіт проекту, взаємозв'язок цих робіт, терміни їх виконання, тривалість робіт. Також календарний план визначає ресурси, необхідні для успішного виконання проекту.

На рис. 3.1 представлено декомпозицію робіт проекту (WBS). Дана декомпозиція містить перелік всіх робіт проекту, а також їх тривалість у часі. З даної діаграми видно послідовність виконання робіт, а також оцінки тривалості виконання всіх робіт.

Оцінки тривалості дають можливість показати дати завершення роботи над проектом. Графік робіт здвигається по мірі роботи над проектом, згідно змін у виконаних роботах.

Діаграма Ганта, що містить послідовність виконання робіт, зображена на рис. 3.2. Тому дана діаграма забезпечує можливість оцінити взаємозв’язок між роботами проекту, необхідні для виконання таких робіт ресурси.

	Длительно	Начало	Окончание	Названия ресурсов
Формування вимог	**8 дней**	**Пн 08.11.21**	**Ср 17.11.21**	
Дослідження предметної областті	2 дней	Пн 08.11.21	Вт 09.11.21	Аналітик;Консультант з предметної області
Формування вимог до проекту	2 дней	Ср 10.11.21	Чт 11.11.21	Менеджер проекту
Виконання необхідних	2 дней	Пт 12.11.21	Пн 15.11.21	Аналітик
Розробка концепції та постановка задачі	3 дней	Пн 15.11.21	Ср 17.11.21	Аналітик;Консультант з предметної області
Розробка технічного завдання	3 дней	Чт 18.11.21	Пн 22.11.21	Менеджер проекту
Проектування	4 дней	Вт 23.11.21	Пт 26.11.21	Програміст
Розробка модулю визна	5 дней	Пн 29.11.21	Пт 03.12.21	Програміст
Тестування	2 дней	Пн 06.12.21	Вт 07.12.21	Тестувальник
Інтеграція	2 дней	Ср 08.12.21	Чт 09.12.21	Програміст
Розробка звіту	1 день	Пт 10.12.21	Пт 10.12.21	Менеджер проекту

Рисунок 3.1 – Декомпозиція робіт проекту

Діаграма Ганта задає розклад робіт у вигляді календарного робочого плану.

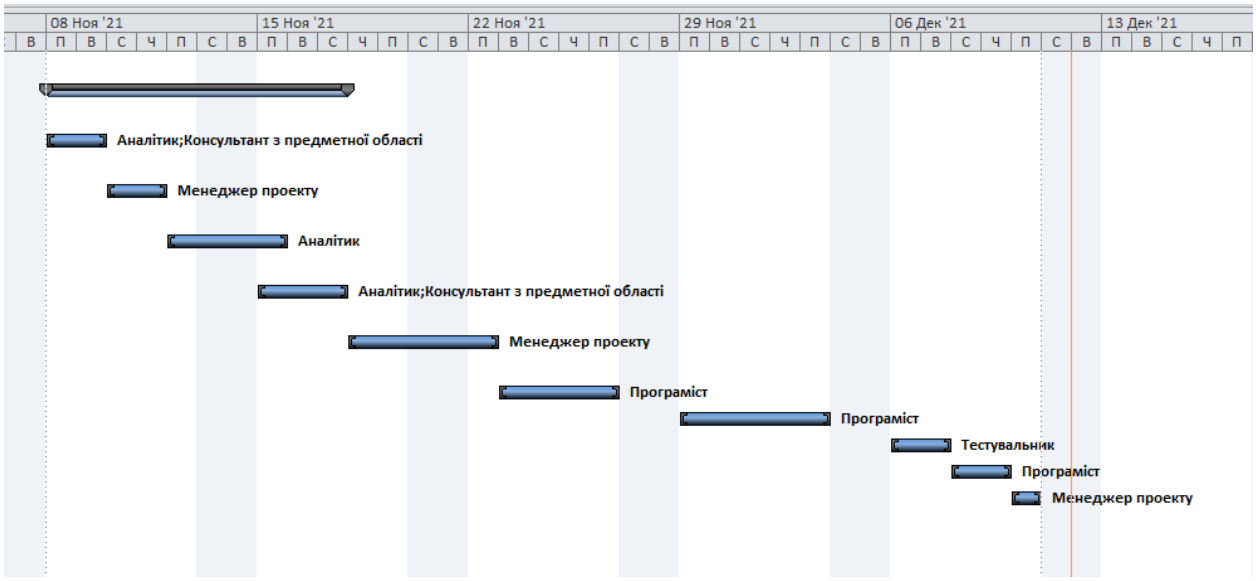

Рисунок 3.2 – Діаграма Ганта

Тому використання даної діаграми при плануванні проекту дає можливість зменшити кількість подальших змін у процесі реалізації проектних заходів.

Хоча діаграма Ганта й показує послідовність запланованих операцій по вирішенню задачі проекту задачі, але вона не відображає ресурсоємність та області дії цих робіт.

Важлива перевага даної діаграми полягає в тому, що вона дає можливість визуально відобразити критичний шлях, тобто найдовший послідовність операцій в проекті. Аналіз критичного шляху дає можливість встановити послідовність завдань, порушення регламентів часу яких приводить до порушень строків завершення всього проекту. Тому критичний шлях проекту, або Critical Path Method (CPM) є важливим елементом проектного управління. Традиційно метод критичного шляху (МКП) у комплексі із системою PERT або її аналогами знайшов застосування при мережному плануванні і управлінні.

Організаційна структура проекту складається з менеджера проекту, у підпорядкування якого є девелопер (програміст) та тестувальник. Створення команди проекту як єдиного робочого колективу пояснює важливість розробки організаційної структури. Цей колектив є тимчасовим, може складатися зі спеціалістів із різних підрозділів підприємства та включати як фахівців з боку Виконавця, так і спеціалістів з боку Замовника. Для кожного спеціаліста нового колективу визначаються їх ролі, функції у проекті, а також відповідальність і повноваження, задаються правила їх взаємодії. Останні представляються організаційною схемою, яка показує ієрархічну упорядкованість команди.

RBS – таблиця (таблиця використання ресурсів) відображає опис усіх ресурсів проекту (рис.3.3). Графік ресурсів містить інформацію про всі аспекти використання ресурсів. Зокрема, задається робота, завантаження, додаткові можливості використання.

Ця таблиця використовується тому, що дає можливість порівнювати використання декількох ресурсів, а також групи ресурсів, тобто бути використаною як аналітичний інструмент.

Подробности	Ноябрь 2021										Декабрь 2021				
	01	04	07	10	13	16	19	22	25	28	01	04	07	10	13
Трудозатр.				16ч		8ч	8ч	8ч						8ч	
Трудозатр.				16ч											
Трудозатр.						8ч	8ч	8ч							
Трудозатр.														8ч	
Трудозатр.								16ч	16ч	16ч	24ч		16ч		
Трудозатр.								16ч	16ч						
Трудозатр.										16ч	24ч				
Трудозатр.													16ч		
Трудозатр.												8ч	8ч		
Трудозатр.												8ч	8ч		
Трудозатр.			16ч	8ч	16ч	16ч									
Трудозатр.			16ч												
Трудозатр.				8ч	8ч										
Трудозатр.					8ч	16ч									
Трудозатр.			16ч		8ч	16ч									
Трудозатр.			16ч												
Трудозатр.					8ч	16ч									

Рисунок 3.3 – Таблиця використання ресурсів

Представлена на рис. 3.3 таблиця використання ресурсів містить представлену у годинах зайнятість спеціалістів, що були задіяні у виконанні проекту для методу визначення пріоритетів.

Дана таблиця забезпечує розуміння та дає можливість структурувати усі етапи розробки, деталізувати затрати часу на виконання проекту, а також визначити повний спектр робіт та його розподіл серед виконавців даного проекту.

У лівій частині таблиці задається назва ресурсу. У правій частині наведено дані по виконанню робіт по дням та годинам, які забезпечуються цим ресурсом.

Таблицю розподілу робіт наведено на рис.3.4.

Режим задачи	Название задачи	Трудозатраты	Длительность	Начало	Окончание	До
	Формування вимог	**112 ч**	**8 дней**	**Пн 08.11.21**	**Ср 17.11.21**	
	Дослідження предметної області	32 ч	2 дней	Пн 08.11.21	Вт 09.11.21	
	Аналітик	*16 ч*		*Пн 08.11.21*	*Вт 09.11.21*	
	Консультант з предметної області	*16 ч*		*Пн 08.11.21*	*Вт 09.11.21*	
	Формування вимог до проекту	16 ч	2 дней	Ср 10.11.21	Чт 11.11.21	
	Менеджер проекту	*16 ч*		*Ср 10.11.21*	*Чт 11.11.21*	
	Виконання необхідних науково-дослідних робіт	16 ч	2 дней	Пт 12.11.21	Пн 15.11.21	
	Аналітик	*16 ч*		*Пт 12.11.21*	*Пн 15.11.21*	
	Розробка концепції та постановка задачі	48 ч	3 дней	Пн 15.11.21	Ср 17.11.21	
	Аналітик	*24 ч*		*Пн 15.11.21*	*Ср 17.11.21*	
	Консультант з предметної області	*24 ч*		*Пн 15.11.21*	*Ср 17.11.21*	
	Розробка технічного завдання	24 ч	3 дней	Чт 18.11.21	Пн 22.11.21	
	Менеджер проекту	*24 ч*		*Чт 18.11.21*	*Пн 22.11.21*	
	Проектування	32 ч	4 дней	Вт 23.11.21	Пт 26.11.21	
	Програміст	*32 ч*		*Вт 23.11.21*	*Пт 26.11.21*	
	Розробка модулю визначення пріоритетів задачі IT проекту	40 ч	5 дней	Пн 29.11.21	Пт 03.12.21	
	Програміст	*40 ч*		*Пн 29.11.21*	*Пт 03.12.21*	
	Тестування	16 ч	2 дней	Пн 06.12.21	Вт 07.12.21	
	Тестувальник	*16 ч*		*Пн 06.12.21*	*Вт 07.12.21*	
	Інтеграція	16 ч	2 дней	Ср 08.12.21	Чт 09.12.21	
	Програміст	*16 ч*		*Ср 08.12.21*	*Чт 09.12.21*	
	Розробка звіту	8 ч	1 день	Пт 10.12.21	Пт 10.12.21	
	Менеджер проекту	*8 ч*		*Пт 10.12.21*	*Пт 10.12.21*	

Рисунок 3.4 – Таблиця розподілу робіт

Дана таблиця дає деталізацію витрат часу на роботи у проекті. Данні про дату виконання а також необхідний час на роботи відображаються в правій частині.

4. ПРАКТИЧНЕ ВИКОРИСТАННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1 Розробка модулю уточнення пріоритетів на основі моделі Кано

Даний програмний модуль є з веб-додатком на мові Java, який інтегрується із існуючою системою підтримки Scrum- рішень ScrumDsk.

Дана система містить базову функціональність, що підтримує роботу із Scrum – методологією. Модуль використовує дані з цієї системи і уточнює рейтинги на основі введення додаткової інформації.

Дану мову для розробки модулю уточнення пріоритетів було вибрано по тій причині, що на сьогодні вона підтримується багатьма платформами, що повністю задовольняє вимоги до використання даного додатку.

Мова програмування, що розглядається має такі переваги, які є важливим при реалізації даного модулю:

– характеризується мультиплатформеністю, що дозволяє використати програмне забезпечення на цій мові для застосування всіх основних вживаних програмно-апаратних платформах; мова орієнтована на роботу в середовищі всіх відомих операційних систем;

– програмне забезпечення на мові Java може бути використано в клієнт-серверній архітектурі за умови, що на сервері встановлено JRE – систему роботи в реальному часі, Java Runtime Environment;

– завдяки використанню JRE програмне забезпечення Java виконується на окремій віртуальній машині JVM; це забезпечує безпеку, тому що програмне забезпечення є ізольованим від ресурсів сервера (апаратна платформа) та операційної системи;

– мова програмування розрахована на всі основні веб-браузери, оскільки вони мають можливість підтримки роботи програмних додатків, що були написані на мові Java.

Розроблений модуль визначення пріоритетів отримує вхідні дані за результатами опитування в гугл -формах.

Фрагмент вхідних для одного завдання даних наведено на рис. 4.1

Timestamp	user id	Ваше відношення, якщо буде реалізовано	Якщо не буде реалізовано	Наскільки важливо
11/19/2021 16:53:38	user1	Все рівно	Так і має бути	Критично важливе
11/19/2021 16:53:38	user2	Подобається	Не подобається	Дуже важливо
11/19/2021 16:53:38	user3	Подобається	Подобається	Було б непогано
11/19/2021 16:53:38	user4	Подобається	Все рівно	Дуже важливо
11/19/2021 16:53:38	user13	Подобається	Подобається	Можливо, буде корисним
11/19/2021 16:53:38	user14	Подобається	Подобається	Неважливо
11/19/2021 16:53:38	user15	Подобається	З цим якось можна жити	Було б непогано
11/19/2021 16:53:38	user17	Подобається	Не подобається	Було б непогано
11/20/2021 15:50:33	user18	Подобається	Не подобається	Було б непогано
11/20/2021 15:50:33	user19	Подобається	Не подобається	Було б непогано
11/20/2021 15:50:33	user21	Подобається	З цим якось можна жити	Було б непогано

Рисунок 4.1 – Фрагмент вхідних даних методу

4.2 Реалізація та перевірка удосконаленого методу

Приклад виконання першого етапу - оцінки задач наведено на рис. 4.2.

Оцінка цих вхідних даних за моделлю Карно приводить до результату, який дає можливість упорядкувати задачі по таким групам: обов'язкові, лінійні, привабливі, непотрібні.

M	O	A	R	P+Q	Total
0,07	0,24	0,28	0,03	0,38	1,00
0,14	0,14	0,38	0,00	0,34	1,00
0,07	0,31	0,21	0,07	0,34	1,00
0,13	0,38	0,13	0,02	0,33	1,00
0,03	0,31	0,31	0,00	0,34	1,00

Рисунок 4.2 – Приклад базової оцінка задач за моделлю Карно

На рис. 4.2 використані такі позначення категорій важливості задач користувача [24]:

A – приваблива (attractive);

O – лінійна (one–dimensional);

M – наявність є обов’язковою (must–be);

R– непотрібні задачі (reverse);

Червоним шрифтом виділено категорії, які не використовуються у подальших кроках методу:

Q – сумнівний/суперечливий результат (questionable result);

I – користувачеві байдуже (indifferent);

На даному рисунку зеленим виділені комірки таблиці, що мають найбільші значення показників g_j^+, отриманих в результаті опитування користувачів, а червоним – найменші значення g_j^-.

Використання цих показників дає можливість уточнити рейтинг задачі: чим більше відхилення між ними, тим більше має змінитися рейтинг. Якщо більшості користувачів така функціональність потрібна, і лише меншості – не

потрібна, то тоді доцільно підвищити рейтинг, щоб пришвидшити реалізацію цієї функціональності.

Модуль уточнення пріоритетів забезпечує можливість вводу даних по задачам (user story) з виставленням пріоритетів для кожної із задач.

Форму вводу даних представлено на рис. 4.3.

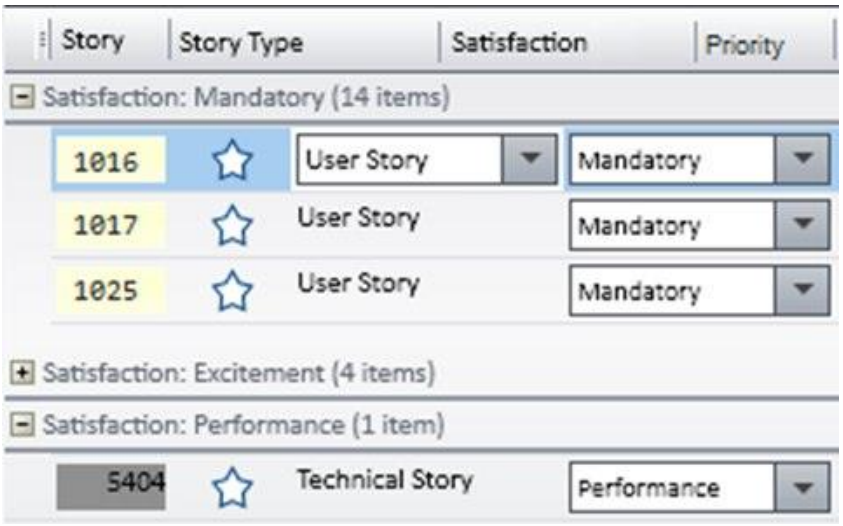

Рисунок 4.3 – Форма для вводу даних про user story

Кожна історія отримує ідентифікатор, який в подальшому використовується при відображанні скоригованого рейтингу історії.

Звіт по попереднім пріоритетам історій (задач IT проекту)представлено на рис. 4.4.

ID	User Story	Backlog Priority	Estimated Effort
1000	Search for resumes by keywrds	1	3
1001	Enter resumes online	2	5
1002	Post a job opening	3	2
1008	Add social network	4	3
1009	Post job opening to my social networks	5	20
1016	Add recruter profile	6	5
1017	Add rating to a recruter	7	8
1025	Revew skill suggestions	8	13
1026	Find job openings that math my skills	9	3
1003	Find resumes with skills that math a job opening	10	3

Рисунок 4.4 – Перелік пріоритетів задач IT проекту (user story)

За результатами уточнення пріоритети історій змінюються. Візуалізація уточнених пріоритетів представлена на рис. 4.5.

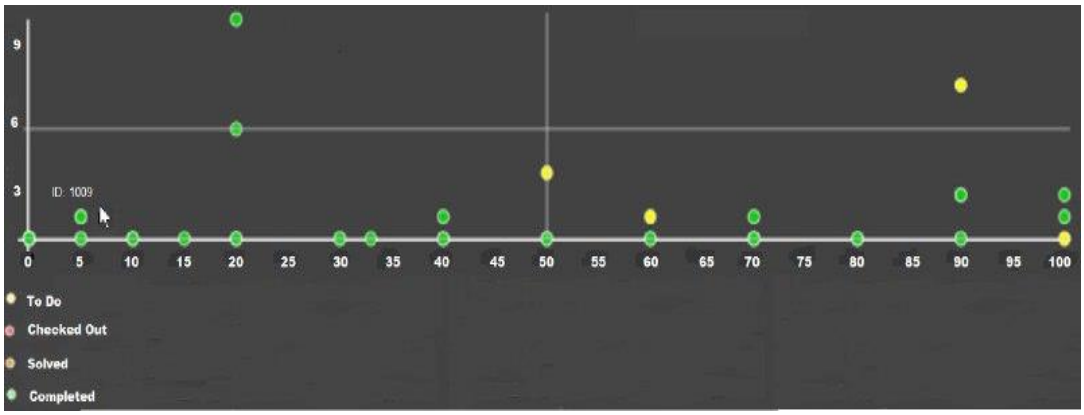

Рисунок 4.5 – Уточнення пріоритетів задач

Наприклад, на рис. 4.5 показано підвищення пріоритетів задачі з ІД 1009 (базове значення пріоритету дорівнює 5).

Запропонований метод дає можливість врахувати зміни у пріоритетах задач з урахуванням змін у потребах користувачів продукту, що розробляється за гнучкою методологією в рамках IT проекту.

На практиці використання уточнених пріоритетів дає можливість скоротити час, коли будуть виконані ключові вимоги споживачів.

ВИСНОВКИ

В кваліфікаційні роботі проведено дослідження проблеми підтримки удосконалення процесу розробки ІТ проекту.

Проведений в кваліфікаційній роботі аналіз показав, що існуючі методи підтримки рішень в ІТ проектах використовують гнучку технологію розробки. За цією технологією використовується уточнення послідовності робіт шляхом зміни пріоритетів для задач ІТ проекту.

Пріоритети виставляються на основі оцінок користувачів, оцінок власника проекту.

Однак необхідність постійного удосконалення процесу розробки при зміні оцінок користувачів, затримках у розробці задач потребують уточнення пріоритетів на основі інтеграції даних підходів.

В роботі вирішено задачі структуризації процесу підтримки прийняття рішень щодо удосконалення процесу розробки ІТ проекту; удосконалення методу визначення пріоритетів розробки задач ІТ проекту з використанням моделі задоволення потреб Кано; планування проекту розробки методу визначення пріоритетів розробки задач ІТ проекту; експериментальної перевірки отриманого удосконаленого методу визначення пріоритетів розробки задач ІТ проекту.

Удосконалено метод визначення пріоритетів розробки задач ІТ проекту з використанням моделі задоволення потреб Кано шляхом уточнення пріоритетів на основі різниці балів користувачів щодо наявності та відсутності таких задач у проекті.

Метод додатково враховує бали користувачів щодо задач та затримки при розробці проекту, що дає можливість динамічно змінювати процес розробки на основі уточнення послідовності реалізації задач. На практиці метод забезпечує можливість упорядкування задач за пріоритетами на

динамічного основі уточнення цих пріоритетів з урахуванням даних щодо поточних вимог користувачів.

Результати магістерської роботи представлені у матеріалах 25-го міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті» 2021р. опубліковані тези доповіді за темою дослідження.