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 процесних методів монітирингу Agile-проєктів_____
_____
(тема)

Виконав:
студент _2_ курсу, групи _УПГІТм-22-3___
____Сосницький Олександр Вадимович____
(прізвище, ім'я, по батькові)
Спеціальність ____122 Комп'ютерні_____
_____ науки_____
(код і повна назва спеціальності)
Тип програми ____ освітньо-наукова_____
(освітньо-професійна або освітньо-наукова)
Освітня програма _ Управління проєктами в галузі інформаційних технологій_____
(повна назва освітньої програми)

Керівник _ проф. каф.ІУС_Сергій ЧАЛИЙ__
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри _____ _ Костянтин ПЕТРОВ__
(підпис) (власне ім'я, прізвище)

2024 р.

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук_____

Кафедра _____ Інформаційних управляючих систем_____

Рівень вищої освіти _____другий (магістерський)_____

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

Тип програми _____ освітньо-наукова_____
(освітньо-професійна або освітньо-наукова)

Освітня програма_ Управління проєктами в галузі інформаційних технологій
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_01__» __квітня__ 20 24__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові_____Сосницький Олександр Вадимович_____
(прізвище, ім'я, по батькові)

1. Тема роботи_Дослідження процесних методів моніторингу Agile-проєктів_____
_____
затверджена наказом університету від _01 квітня_2024_ р. № _258 СТ_
2. Термін подання студентом роботи до екзаменаційної комісії __01_ ___06_____ 2024_ р.
3. Вихідні дані до роботи _Інформація щодо стану технологій у обраній предметній галузі; аналітичні та наукові статті методів моніторингу Agile-проектів; аналітичні та наукові статті методів процес-майнингу; деталі використання обраних методів процес-майнингу у_____ обраній_ предметній галузі_____
_____
_____
_____
4. Перелік питань, що потрібно опрацювати в роботі дослідити предметну галузь моніторингу Agile-проектів; дослідити методи моніторингу; дослідити використання методів процес-майнингу для моніторингу Agile-проектів; сформулювати метод вирішення для поставленої задачі дослідження; сформувати план реалізації експериментального методу для рішення поставленої задачі; дослідження для розробки, розробка та експериментальна перевірка методу для вирішення поставленої задачі

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	01.03.2024-15.03.2024	Виконано
2	Дослідження технологій предметної галузі	16.03.2024-31.03.2024	Виконано
3	Постановка задачі дослідження	01.04.2024-15.04.2024	Виконано
4	Теоретичне дослідження обраних технологій	16.04.2024-21.04.2024	Виконано
5	Практичне дослідження обраний технологій	22.04.2024-30.04.2024	Виконано
6	Розробка тестового прототипу	22.04.2024-08.05.2024	Виконано
7	Аналіз недоліків та переваг тестового прототипу	01.05.2024-08.05.2024	Виконано
8	Моделювання висновків за виконаною роботою	08.05.2024-16.05.2024	Виконано
9	Написання пояснювальної записки	01.05.2024-28.05.2024	Виконано

Дата видачі завдання _01_ __квітня__2024 р.

Студент _____
(підпис)

Керівник роботи _____ проф. каф.ІУС_Сергій ЧАЛИЙ
(підпис) (власне ім’я, прізвище)

РЕФЕРАТ

Робота містить: 85 с., 38 рис., 9 табл., 5 фор., 1 дод, 32 джерела.

АНАЛІЗ, AGILE-ПРОЕКТИ, ПРОЦЕС-МАЙНІНГ, SCRUM, KANBAN, SCRUMBAN, ЖУРНАЛ ПОДІЙ, УПРАВЛІННЯ ПРОЕКТАМИ, МОНІТОРИНГ, ІТ-ПРОЕКТИ, МЕТОДОЛОГІЇ, АНАЛІЗ ДАНИХ, МОДЕЛЮВАННЯ ПРОЦЕСІВ, ЕФЕКТИВНІСТЬ, ОПТИМІЗАЦІЯ ПРОЦЕСІВ, ВІЗУАЛІЗАЦІЯ, ІНСТРУМЕНТИ ПРОЦЕС-МАЙНІНГУ, ALPHA MINER.

Об’єкт дослідження: процеси моніторингу проектів розробки програмного забезпечення.

Ціль роботи: дослідження процесних методів моніторингу в Agile-проектах, планування та пропонування практичної реалізації для покращення моніторингу Agile-проектів інструментами процес-майнингу.

Методи дослідження: дослідити предметну галузь моніторингу Agile-проектів, аналіз існуючих методів моніторингу, формування проблеми дослідження, розробка методу для покращення моніторингу з використанням методів процес-майнингу, планування проекту імплементації рішення, формування проекту, опис основних етапів проекту, розробка детального плану проекту, опис інструментарію, тестування методу моніторингу з використанням інструментів процес-майнингу для Agile-проектів.

ABSTRACT

Explanatory note: 85 p., 38 fig., 9 tab., 5 for., 1 ann., 32 ref.

ANALYSIS, AGILE PROJECTS, PROCESS MINING, SCRUM, KANBAN, SCRUMBAN, EVENT LOG, PROJECT MANAGEMENT, MONITORING, IT PROJECTS, METHODOLOGIES, DATA ANALYSIS, PROCESS MODELING, EFFICIENCY, PROCESS OPTIMIZATION, VISUALIZATION, PROCESS MINING TOOLS, ALPHA MINER.

The object of the research is processes of monitoring software development projects.

The aim of the study is to research process methods of monitoring in Agile projects, plan and propose practical implementation to improve Agile project monitoring using process mining tools.

The research methods are as follows: research the domain of Agile project monitoring, analyze existing monitoring methods, formulate the research problem, develop a method to improve monitoring using process mining methods, plan the implementation project, form the project, describe the main project stages, develop a detailed project plan, describe the tools, test the monitoring method using process mining tools for Agile projects.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

WIP – Work in Progress (Робота у процесі)

CT – Cycle Time (Цикл часу)

LT – Lead Time (Час виконання)

TiS - Time in Status (Час у статусі)

PM - Project Manager (Менеджер проекту)

PR - Pull Request (Запит на злиття)

CR - Code Review (Перегляд коду)

ВСТУП

У сучасному світі управління проектами зіштовхується з викликами швидких змін технологій та вимог ринку, що потребує від компаній високої адаптивності та гнучкості. Agile-методології, такі як Scrum та Kanban, виникли як відповідь на ці виклики, пропонуючи альтернативу традиційному каскадному підходу до управління проектами. Вони орієнтовані на спрощення процесу розробки, покращення командної роботи та забезпечення можливості швидко реагувати на зміни. Проте, ефективність будь-якої методології значною мірою залежить від якості моніторингу та аналізу проектів, які здійснюються в рамках цих підходів.

Ця дипломна робота зосереджена на дослідженні процесних методів моніторингу Agile-проектів, з метою виявлення та аналізу ключових аспектів, які сприяють або заважають ефективному управлінню проектами. Вибір теми зумовлений бажанням глибше зрозуміти, як гнучкі методи можуть покращити процеси в компаніях, що працюють в динамічних умовах і постійно стикаються зі змінами у своїх проектах.

У першому розділі роботи розглядаються теоретичні аспекти Agile-методологій, їхні основні принципи та інструменти. Другий розділ присвячений глибокому аналізу методів моніторингу, що використовуються в компаніях для контролю та адаптації проектних процесів. У третьому розділі пропонуються шляхи вдосконалення процесів моніторингу на основі отриманих даних та власних досліджень. Завершується робота висновками, які підсумовують основні знахідки дослідження та визначають можливі напрямки подальших досліджень у даній області.

1 ДОСЛІДЖЕННЯ ПРОСЕЦНИХ МОНИТОРИНГУ AGILE ПРОЕКТІВ

1.1 Порівняльний аналіз підходів до управління IT-проектами

Існують основні підходи до управління IT-проектами, серед них є класичні підходи, такі як водоспадна модель (Waterfall), а також гнучкі методи (Agile).

Водоспадна модель (Waterfall). Водоспадна модель є одним із найстаріших та найбільш класичних методів управління проектами. Вона передбачає лінійний та послідовний процес, де кожен етап проекту має бути завершений перед початком наступного. Етапи включають збір вимог, проектування, реалізацію, тестування, впровадження та підтримку.

Переваги водоспадної моделі:

– чітко визначені етапи та послідовність дій;

– легкість у плануванні та управлінні завдяки детальному опису кожного етапу;

– зрозумілість для всіх учасників проекту через наявність чіткої документації.

Недоліки водоспадної моделі:

– високий ризик затримок через необхідність завершення кожного етапу перед переходом до наступного;

– відсутність гнучкості та можливості внесення змін після завершення певних етапів;

– проблеми з адаптацією до змін у вимогах клієнта або ринку.

Таблиця 1.1 – Порівняння підходів до управління проектами

Критерій	Традиційний підхід (Waterfall)	Ітераційний підхід	Інкрементальний підхід	Гнучкий підхід (Agile)
Структура процесу	Лінійна, послідовна (планування, розробка, тестування, впровадження)	Повторювані цикли з повним набором етапів в кожній ітерації	Послідовне нарощування функціоналу	Ітеративна, гнучка (постійні зміни, ітерації)
Планування	Детальне, на початку проекту	Планування на початку кожної ітерації	Планування на початку кожного інкременту	Гнучке, адаптивне, постійно оновлюється
Вимоги	Визначаються на початку, фіксовані	Визначаються на початку кожної ітерації	Визначаються на початку проекту, можуть змінюватися з кожним інкрементом	Можуть змінюватися протягом проекту
Комунікація	Формальна, документована	Регулярні зустрічі для оцінки та планування	Регулярні зустрічі для оцінки та планування	Неформальна, часті зустрічі
Контроль	Суворий контроль, контроль якості на кожному етапі	Оцінка та контроль після кожної ітерації	Оцінка та контроль після кожного інкременту	Самоорганізація, контроль якості постійно
Гнучкість	Низька, важко внести зміни після початку	Середня, зміни можливі між ітераціями	Середня, зміни можливі між інкрементами	Висока, легко адаптуватися до змін
Оцінка прогресу	За завершеними етапами (між етапами)	Після кожної ітерації	Після кожного інкременту	Регулярно, за результатами кожної ітерації

Продовження таблиці 1.1

Критерій	Традиційний підхід (Waterfall)	Ітераційний підхід	Інкрементальний підхід	Гнучкий підхід (Agile)
Ризики	Виявляються на пізніх етапах проекту	Виявляються та усуваються після кожної ітерації	Виявляються та усуваються після кожного інкременту	Виявляються та усуваються постійно
Впровадження	В кінці проекту	Після кожної ітерації (якщо вона завершена)	Після кожного інкременту	Протягом всього проекту, інкрементально
Команда	Ролі чітко визначені, робота в межах функціональних підрозділів	Крос-функціональні команди, взаємодія між членами команди	Крос-функціональні команди, взаємодія між членами команди	Крос-функціональні команди, взаємодія між членами команди
Зворотний зв'язок	Після завершення кожного етапу	Після кожної ітерації	Після кожного інкременту	Постійний зворотний зв'язок, швидкі корекції

Гнучкі методи управління проектами, відомі як Agile, стали надзвичайно популярними в IT-індустрії завдяки своїй здатності швидко адаптуватися до змін та забезпечувати високу якість продукту. Agile включає кілька методологій, таких як Scrum, Kanban, Lean, XP (Extreme Programming) та інші.

Основні особливості Agile підходів:

– ітеративний та інкрементальний процес: agile методології використовують короткі, повторювані цикли (спринти), що дозволяє регулярно оцінювати та коригувати напрямок розвитку проекту;

– фокус на командну роботу: agile підходи акцентують увагу на співпраці та комунікації всередині команди, а також із зацікавленими сторонами проекту;

– гнучкість і адаптивність: agile дозволяє швидко реагувати на зміни в вимогах клієнта або ринку, що зменшує ризики невідповідності кінцевого продукту очікуванням;

– зворотний зв'язок та постійне вдосконалення: agile методології передбачають регулярні зустрічі та оцінки (демо, ретроспективи), що дозволяє постійно вдосконалювати процеси та продукт.

Таблиця 1.2 – Порівняння гнучких підходів до управління проектами

Критерій	Scrum	Kanban	Scrumban
Структура процесу	Ітеративна, структурована (спринти, ретроспективи, щоденні зустрічі)	Неперервний потік, фокус на візуалізації роботи та управлінні потоком	Комбінація ітераційного підходу Scrum з гнучкістю Kanban
Планування	Планування на початку кожного спринту	Поступове планування, фокус на пріоритетах та поточній роботі	Планування на початку спринту з можливістю адаптації у реальному часі
Тривалість циклів	Фіксовані спринти (зазвичай 1-4 тижні)	Немає фіксованих циклів, безперервний потік роботи	Поєднання фіксованих спринтів з безперервним потоком роботи
Вимоги	Визначаються та уточнюються на початку кожного спринту	Визначаються постійно, пріоритизуються в міру надходження	Визначаються та уточнюються на початку кожного спринту, можливі зміни

Продовження таблиці 1.2

Критерій	Scrum	Kanban	Scrumban
Комунікація	Щоденні stand-up зустрічі, регулярні ретроспективи	Регулярні зустрічі для обговорення роботи та прогресу	Щоденні stand-up зустрічі, регулярні ретроспективи, зустрічі для управління потоком
Контроль	Самоорганізація команди, Scrum Master допомагає усувати перешкоди	Самоорганізація команди, фокус на управлінні потоком	Самоорганізація команди, комбінація ролей Scrum і управління потоком
Гнучкість	Висока, адаптація до змін кожного спринту	Дуже висока, можливість змін у будь-який момент	Висока, поєднання гнучкості Kanban з структурованістю Scrum
Оцінка прогресу	Після кожного спринту (Sprint Review)	Постійний моніторинг прогресу через візуалізацію на дошці	Після кожного спринту, постійний моніторинг через дошку
Ризики	Виявляються та усуваються протягом кожного спринту	Виявляються та усуваються постійно	Виявляються та усуваються протягом спринту та в процесі
Впровадження	Після кожного спринту, інкрементально	Постійно, по мірі готовності роботи	Після кожного спринту, постійно
Команда	Крос-функціональні команди, ролі Scrum Master, Product Owner	Крос-функціональні команди, ролі можуть бути гнучкими	Крос-функціональні команди, комбінація ролей Scrum і Kanban

Продовження таблиці 1.2

Критерій	Scrum	Kanban	Scrumban
Зворотний зв'язок	Після кожного спринту (Sprint Review та Retrospective)	Постійний зворотний зв'язок через щоденні зустрічі та візуалізацію на дошці	Постійний зворотний зв'язок через щоденні зустрічі, Sprint Review та візуалізацію на дошці
Документація	Мінімум документації, акцент на робочому продукті	Мінімум документації, фокус на візуалізації роботи	Мінімум документації, поєднання акценту на продукті та візуалізації

Scrum є однією з найпопулярніших методологій agile, яка включає визначені ролі (Scrum Master, Product Owner, команда розробників), артефакти (Product Backlog, Sprint Backlog) та події (Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective).

Переваги Scrum:

– чітка структура та визначені ролі;

– регулярні спринти та зворотний зв'язок сприяють високій адаптивності;

– підвищена прозорість та контроль за прогресом проекту.

Недоліки Scrum:

– потреба в чіткому розподілі ролей та функцій;

– складність у великих проектах через необхідність координації між кількома командами.

Kanban фокусується на візуалізації процесу роботи та управлінні потоками завдань через використання Kanban-дошок. Основна мета – забезпечити постійний потік роботи та уникнути затримок.

Переваги Kanban:

– легкість впровадження та використання;

– висока гнучкість та адаптивність до змін;

– прозорість процесів та управління потоком завдань.

Недоліки Kanban:

– відсутність чіткої структури та визначених ролей;

– можливість затримок через недостатню координацію між командами.

Scrumban є комбінацією методологій Scrum і Kanban, яка поєднує переваги обох підходів для забезпечення гнучкості та ефективності в управлінні проектами. Цей підхід використовує структуру та ритуали Scrum для планування та виконання спринтів, водночас застосовуючи візуалізацію та управління потоками завдань за допомогою Kanban-дошок.

Переваги Scrumban:

– легкість впровадження та адаптації до існуючих процесів;

– поєднання структури Scrum з гнучкістю Kanban;

– можливість покращення процесів завдяки постійному аналізу та оптимізації;

– ефективне управління потоками завдань через візуалізацію на Kanban-дошках.

Недоліки Kanban:

– складність у визначенні чітких ролей та обов'язків;

– можливість виникнення хаосу через відсутність чітко визначених меж спринтів;

– потреба у постійному моніторингу та налаштуванні процесів для уникнення перевантаження команди.

Узагальнюючи, кожен підхід до управління IT-проектами має свої особливості, переваги та недоліки. Вибір методу залежить від специфіки проекту, команди та вимог клієнта. Agile методології, такі як Scrum, Kanban та Scrumban, є найбільш гнучкими та адаптивними, що робить їх популярними у сучасному IT-середовищі. Кожен із цих підходів має свої унікальні риси та можливості, які дозволяють командам ефективно планувати, виконувати та моніторити свої проекти.

Scrum є однією з найбільш розповсюджених Agile методологій. Він зосереджений на розбитті роботи на короткі, чітко визначені інтервали часу, відомі як спринти, тривалістю від одного до чотирьох тижнів. Основні ролі в Scrum включають Product Owner, Scrum Master та команду розробників, що забезпечує чітку структуру та розподіл обов'язків. Регулярні зустрічі, такі як щоденні Stand-up, Sprint Planning, Sprint Review та Sprint Retrospective, сприяють прозорості процесів та постійному вдосконаленню.

Рисунок 1.1 – Скрам процес

Kanban, з іншого боку, фокусується на візуалізації процесу роботи та управлінні потоками завдань через використання Kanban-дошок. Основна мета – забезпечити постійний потік роботи та уникнути затримок. Завдання поділяються на кілька категорій, таких як To Do, In Progress, In Review та Done, що дозволяє команді бачити стан кожного завдання в реальному часі. Kanban-дошка допомагає ідентифікувати вузькі місця та вживати заходів для їх усунення. Kanban є дуже гнучким і може бути легко адаптований до будь-яких змін у вимогах проекту. Проте, відсутність чіткої структури та визначених ролей може призводити до недостатньої координації між членами команди та затримок у виконанні завдань.

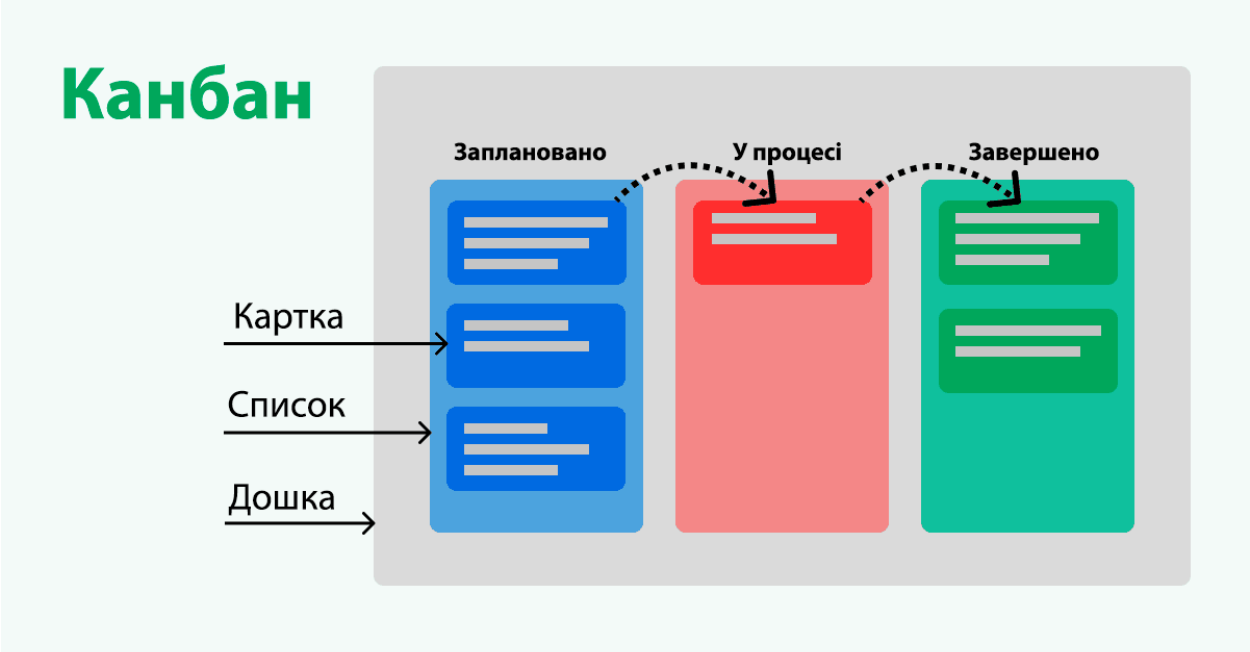


Рисунок 1.2 – Канбан дошка

Scrumban є гібридною методологією, що поєднує елементи Scrum і Kanban, щоб отримати найкраще з обох підходів. Цей підхід використовує структуру та ритуали Scrum для планування та виконання спринтів, водночас застосовуючи візуалізацію та управління потоками завдань за допомогою Kanban-дошок. Scrumban дозволяє командам зберігати чіткість структури та

процесів, характерну для Scrum, водночас забезпечуючи гнучкість та адаптивність, властиві Kanban. Основні переваги Scrumban включають легкість впровадження та адаптації до існуючих процесів, поєднання структури Scrum з гнучкістю Kanban, а також можливість покращення процесів завдяки постійному аналізу та оптимізації. Однак, цей підхід може мати складнощі у визначенні чітких ролей та обов'язків, а також потребує постійного моніторингу та налаштування процесів для уникнення перевантаження команди.

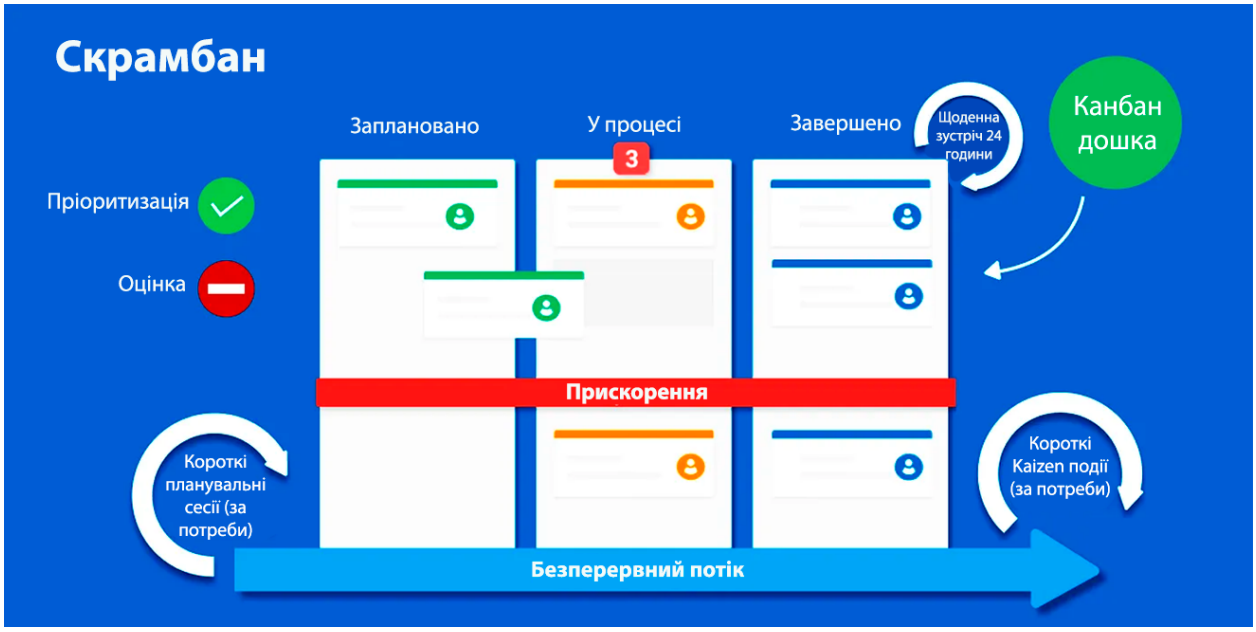


Рисунок 1.3 – Скрамбан процес

Scrumban забезпечує оптимальне поєднання чіткості та структури, які характерні для Scrum, з гнучкістю та візуалізацією процесів, що надаються Kanban. У цьому підході команди продовжують працювати з визначеними спринтами, але водночас використовують Kanban-дошки для відстеження прогресу завдань у реальному часі. Це дає можливість більш гнучко керувати робочими потоками, швидко реагувати на зміни та оптимізувати процеси на основі актуальної інформації.

Основні переваги Scrumban включають легкість впровадження та адаптації до існуючих процесів. Цей підхід дозволяє командам, які вже працюють за Scrum, поступово інтегрувати елементи Kanban, зберігаючи при цьому знайому структуру та ритуали. Наприклад, команда може продовжувати проводити спринти та зустрічі Sprint Planning, Sprint Review та Sprint Retrospective, але водночас використовувати Kanban-дошки для візуалізації завдань та покращення управління робочими потоками.

Scrumban поєднує структуру Scrum з гнучкістю Kanban, що дозволяє командам швидко адаптуватися до змін. У випадку, коли пріоритети змінюються або з'являються нові завдання, команда може легко інтегрувати їх у поточний робочий процес без необхідності чекати завершення спринту. Це забезпечує більш динамічне управління проектами та підвищує загальну ефективність команди.

Крім того, Scrumban надає можливість покращення процесів завдяки постійному аналізу та оптимізації. Регулярний зворотний зв'язок та аналіз результатів дозволяють командам виявляти проблеми та вузькі місця у процесах, що сприяє їх усуненню та вдосконаленню методів роботи. Використання Kanban-дошок дозволяє більш точно відстежувати виконання завдань, ідентифікувати перешкоди та швидко вживати заходів для їх усунення.

Однак, цей підхід може мати складнощі у визначенні чітких ролей та обов'язків. Наприклад, у Scrum чітко визначені ролі Product Owner, Scrum Master та команда розробників. У Scrumban ці ролі можуть бути менш формалізованими, що може призводити до плутанини та невизначеності у відповідальності за різні аспекти проекту. Це вимагає від команди високого рівня самоорганізації та комунікації, щоб уникнути конфліктів та забезпечити ефективну співпрацю.

Scrumban також потребує постійного моніторингу та налаштування процесів для уникнення перевантаження команди. У Kanban є обмеження на

кількість завдань, які можуть бути в роботі одночасно (Work In Progress - WIP). Ці обмеження допомагають уникнути перевантаження команди та забезпечити стабільний потік роботи. У Scrumban важливо правильно налаштувати ці обмеження та постійно їх переглядати, щоб забезпечити баланс між навантаженням команди та ефективністю виконання завдань.

Важливо зазначити, що Scrumban дозволяє поступово впроваджувати зміни та покращення у процесах, що робить його ідеальним для команд, які прагнуть підвищити свою гнучкість та ефективність без значних змін у поточній структурі роботи. Використання Kanban-дошок забезпечує візуалізацію процесів, що сприяє кращому розумінню стану проекту всіма членами команди та зацікавленими сторонами.

Окрім цього, Scrumban сприяє покращенню командної взаємодії та комунікації. Завдяки регулярним зустрічам та постійному зворотному зв'язку, команди можуть швидко вирішувати проблеми, обговорювати прогрес та вносити необхідні корективи. Це сприяє підвищенню прозорості процесів, покращенню якості роботи та забезпеченню досягнення поставлених цілей.

Таким чином, Scrumban є потужним інструментом для управління проектами, що поєднує переваги двох найпопулярніших Agile методологій - Scrum та Kanban. Він дозволяє командам зберігати чіткість структури та процесів, характерну для Scrum, водночас забезпеч уючи гнучкість та адаптивність, властиві Kanban. Цей підхід дозволяє командам максимально ефективно використовувати свої ресурси, швидко адаптуватися до змін та постійно вдосконалювати свої процеси.

Основні переваги Scrumban включають легкість впровадження та адаптації до існуючих процесів. Наприклад, команди, які вже працюють за Scrum, можуть поступово впроваджувати елементи Kanban, не порушуючи при цьому основну структуру та ритуали Scrum. Це дозволяє забезпечити плавний перехід та мінімізувати ризики, пов'язані з впровадженням нових методологій. Крім того, Scrumban забезпечує можливість постійного

вдосконалення процесів, завдяки регулярному аналізу результатів та впровадженню змін на основі отриманих даних.

Однією з ключових особливостей Scrumban є використання Kanban-дошок для візуалізації робочих процесів. Це дозволяє команді чітко бачити стан кожного завдання, відстежувати його прогрес та швидко реагувати на будь-які зміни. Kanban-дошки допомагають ідентифікувати вузькі місця у процесах та приймати обґрунтовані рішення щодо їх усунення. Візуалізація процесів також сприяє покращенню комунікації всередині команди та підвищенню прозорості проекту для всіх зацікавлених сторін.

Застосування ритуалів Scrum, таких як спринти, зустрічі для планування спринтів, огляди спринтів та ретроспективи, дозволяє зберігати структуру та ритм роботи команди. Це забезпечує чіткість у плануванні та виконанні завдань, а також можливість регулярно отримувати зворотний зв'язок та вдосконалювати процеси. Поєднання цих ритуалів з гнучкістю Kanban дозволяє командам швидко адаптуватися до змін та забезпечити безперервний потік роботи.

Проте, використання Scrumban вимагає постійного моніторингу та налаштування процесів для уникнення перевантаження команди. Важливо правильно налаштувати обмеження на кількість завдань у роботі (WIP), щоб забезпечити баланс між навантаженням команди та ефективністю виконання завдань. Це потребує регулярного аналізу та перегляду встановлених обмежень, щоб забезпечити оптимальну продуктивність команди.

Також слід враховувати, що Scrumban може мати складнощі у визначенні чітких ролей та обов'язків. Наприклад, у Scrum ролі Product Owner, Scrum Master та команди розробників чітко визначені, що забезпечує розподіл обов'язків та відповідальності. У Scrumban ці ролі можуть бути менш формалізованими, що може призводити до плутанини та невизначеності у відповідальності за різні аспекти проекту. Це вимагає від команди високого

рівня самоорганізації та комунікації, щоб уникнути конфліктів та забезпечити ефективну співпрацю.

Scrumban також сприяє покращенню командної взаємодії та комунікації. Завдяки регулярним зустрічам та постійному зворотному зв'язку, команди можуть швидко вирішувати проблеми, обговорювати прогрес та вносити необхідні корективи. Це сприяє підвищенню прозорості процесів, покращенню якості роботи та забезпеченню досягнення поставлених цілей.

У підсумку, Scrumban є потужним інструментом для управління проектами, що поєднує переваги двох найпопулярніших Agile методологій – Scrum та Kanban. Він дозволяє командам зберігати чіткість структури та процесів, характерну для Scrum, водночас забезпечуючи гнучкість та адаптивність, властиві Kanban. Це забезпечує оптимальне поєднання ефективності та гнучкості, що є критично важливим для успішного виконання проектів у сучасному IT-середовищі.

Кожен з цих підходів має свої унікальні переваги, які можуть бути особливо корисними в різних контекстах. Наприклад, Scrum ідеально підходить для проектів, де потрібна чітка структура та регулярний зворотний зв'язок, що допомагає тримати команду сфокусованою та мотивованою. Kanban, з його гнучкістю та простотою, є чудовим вибором для команд, які працюють у середовищі з постійно змінюваними вимогами та де потрібно швидко адаптуватися до нових умов. Scrumban, як компромісний варіант, підходить для команд, які вже працюють за Scrum, але потребують додаткової гнучкості та можливості швидко реагувати на зміни в проектах.

Agile методології в цілому спрямовані на підвищення ефективності командної роботи та забезпечення високої якості продуктів, що розробляються. Вони дозволяють командам швидко адаптуватися до змін, зосереджуватися на пріоритетних завданнях та забезпечувати регулярний зворотний зв'язок, що є критично важливим для успішного виконання проектів у сучасному IT-середовищі. Незалежно від вибору методології, основним

принципом Agile залишається прагнення до безперервного вдосконалення та максимального задоволення потреб клієнтів.

У кінцевому підсумку, вибір методу управління проектами повинен базуватися на специфіці проекту, характеристиках команди та вимогах клієнта. Agile методології, такі як Scrum, Kanban та Scrumban, надають командам інструменти та підходи, які дозволяють ефективно управляти проектами та досягати високих результатів у динамічному та конкурентному IT-середовищі.

1.2 Дослідження процесу моніторингу Agile-проектів

Моніторинг в Agile-проектах є критично важливим для забезпечення успішного виконання завдань, досягнення цілей проекту та підтримки високої ефективності команди. Нижче наведено детальну послідовність кроків для ефективного моніторингу Agile-проектів. Схема процесу моніторингу включає кілька ключових етапів, які забезпечують ефективне управління та контроль над виконанням проектних завдань. Схема цього процесу зображена на рисунку 1.4.

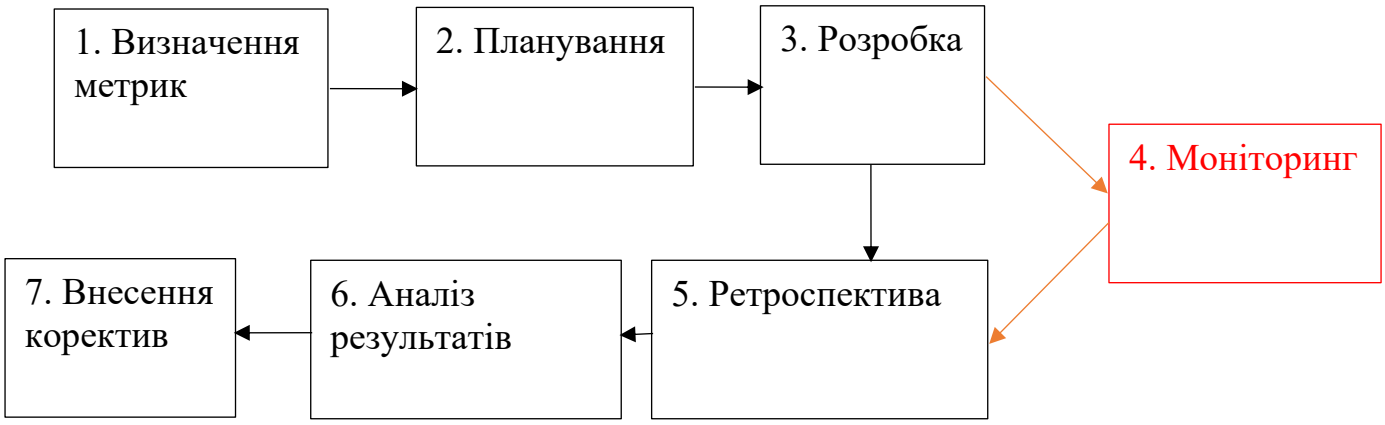


Рисунок 1.4 – Етап моніторингу

На першому етапі необхідно визначити ключові метрики, які будуть використовуватися для оцінки прогресу проекту та ефективності команди. До таких метрик можуть входити:

– Cycle Time (CT): час, що витрачається на завершення завдання з моменту його початку до завершення;

– Lead Time (LT): час, що витрачається на завершення завдання з моменту його створення до завершення.

Планування спринту включає визначення завдань та цілей на наступний спринт. Команда проводить зустріч, на якій обговорюються:

– завдання зі списку Product Backlog, які будуть виконуватися у спринті;

– пріоритетність завдань та їх розподіл між членами команди;

– очікувані результати та цілі спринту.

Щоденні короткі зустрічі (Stand-up) дозволяють команді синхронізувати свої дії, обговорити прогрес та вирішити можливі проблеми. На таких зустрічах обговорюються:

– що було зроблено вчора;

– що планується зробити сьогодні;

– які перешкоди заважають виконанню завдань.

Для відстеження прогресу використовується візуалізація завдань на дошці (Kanban board) або в інструментах управління проектами (Jira, Trello). Завдання на дошці поділяються на кілька колонок:

– To Do (Заплановані);

– In Progress (В роботі);

– In Review (На перевірці);

– Done (Завершені).

Команда постійно відстежує виконання завдань, використовуючи визначені метрики та інструменти візуалізації. Це дозволяє вчасно виявляти

проблеми та коригувати дії для їх вирішення. Збір та аналіз даних з таких інструментів, як GitHub, Jira, GitLab, дозволяє отримати об'єктивну інформацію про прогрес.

Після завершення спринту проводиться оцінка досягнутих результатів. На зустрічі Sprint Review команда представляє виконані завдання та отримує зворотний зв'язок від зацікавлених сторін. Основні аспекти, які розглядаються:

– оцінка виконаних завдань порівняно з запланованими;
– виявлення відхилень та їх причин;
– аналіз використаних метрик для оцінки прогресу.

На завершальному етапі кожного спринту проводиться ретроспектива (Sprint Retrospective). Це зустріч, на якій команда аналізує свої процеси та визначає, що можна покращити у наступних спринтах. Основні питання, які обговорюються:

– що було зроблено добре;
– що можна покращити;
– які дії необхідно вжити для вдосконалення процесів.

На основі результатів ретроспективи команда вносить корективи у процеси розробки та моніторингу. Це можуть бути зміни у методах роботи, використанні інструментів, комунікації або організації завдань. Важливо постійно вдосконалювати процеси для підвищення ефективності та якості роботи.

Моніторинг в Agile-проектах є безперервним процесом, який включає визначення ключових метрик, планування, щоденні зустрічі, візуалізацію прогресу, аналіз результатів, ретроспективи та внесення коректив. Такий підхід забезпечує високу гнучкість, прозорість процесів та постійне вдосконалення, що сприяє успішному виконанню проектів.

Переваги моніторингу у гнучкому підході:

– швидке виявлення проблем: регулярні зустрічі та постійний зворотний зв'язок дозволяють оперативно виявляти та вирішувати проблеми;

– підвищена гнучкість та адаптивність: команда може швидко адаптуватися до змін у вимогах та пріоритетах, що забезпечує відповідність результатів очікуванням замовника;

– постійне вдосконалення: ретроспективи дозволяють команді аналізувати свої процеси та постійно шукати способи їх покращення;

– прозорість та залученість команди: всі члени команди завжди в курсі поточного стану проекту, що сприяє кращій координації та підвищенню командного духу.

Недоліки моніторингу у гнучкому підході:

– можливість виникнення хаосу через постійні зміни: постійні зміни можуть призвести до втрати фокусу та дезорганізації команди;

– складність у забезпеченні прозорості та документації процесів: висока швидкість змін може ускладнити документування та забезпечення прозорості процесів, що може призвести до проблем у відстеженні прогресу та контролі якості.

Для подолання недоліків Agile підходу, таких як можливість виникнення хаосу через постійні зміни та складність у забезпеченні прозорості та документації процесів, доцільно використовувати оброблену інформацію про події процесу розробки. Це дозволяє порівнювати реальний процес розробки із запланованим та отримувати більш точне уявлення про виконані дії.

Порівняння реального процесу розробки з запланованим дозволяє виявити відхилення, вчасно коригувати процеси та підвищувати їх ефективність. Результатом моніторингу буде не лише послідовність даних про виконані дії, але й графове представлення фактичного процесу розробки.

Важливою частиною ефективного моніторингу є використання сучасних інструментів та технологій для збору і аналізу даних. Інтеграція з системами контролю версій, такими як GitHub, дозволяє автоматизувати збір даних про всі події в проекті. Ці дані можна використовувати для побудови детальних графів процесу, які відображають реальний стан справ. Візуалізація таких

графів допомагає команді краще розуміти процеси та виявляти проблемні зони. Окрім GitHub, існують інші інструменти, такі як Jira, Trello та GitLab, які також можуть бути інтегровані для отримання повного зображення про хід проекту.

Сучасні інструменти дозволяють не лише збирати дані, але й аналізувати їх у режимі реального часу. Це забезпечує можливість оперативно реагувати на зміни та виявляти потенційні проблеми ще до їх виникнення. Наприклад, автоматизовані системи сповіщення можуть повідомляти команду про відхилення від плану, затримки або інші критичні події. Використання таких технологій значно підвищує швидкість та ефективність прийняття рішень.

Регулярне проведення ретроспектив та аналізу отриманих даних дозволяє команді постійно вдосконалювати свої методи роботи. Зокрема, використання методів процес-майнінгу допомагає виявляти закономірності та аномалії у процесі розробки, що може сприяти прийняттю більш обґрунтованих рішень щодо подальших дій. Аналізуючи зібрані дані, можна визначити, які етапи процесу потребують покращення, які методи роботи є найефективнішими, а які - менш результативні.

Крім того, важливо враховувати зворотний зв'язок від усіх зацікавлених сторін проекту. Це допомагає не лише виявляти та вирішувати поточні проблеми, але й забезпечувати більш точне планування та прогнозування майбутніх спринтів. Залучення замовників та інших зацікавлених осіб до процесу моніторингу та ретроспективи дозволяє забезпечити, що кінцевий продукт відповідає їхнім очікуванням та вимогам. Спільна робота з клієнтами та іншими стейкхолдерами сприяє формуванню більш точних та реалістичних цілей проекту, що знижує ризики та підвищує задоволеність клієнтів результатами.

Збір та аналіз даних мають бути безперервними процесами. Використання систем моніторингу та аналізу, таких як ProM, дозволяє автоматизувати багато рутинних задач, що дає змогу команді зосередитися на

стратегічних аспектах проекту. Інструменти процес-майнінгу можуть не лише виявляти відхилення, але й пропонувати шляхи їх усунення на основі аналізу попередніх даних.

Постійне вдосконалення процесів є ключовим аспектом Agile-методологій. Це включає не лише корекцію поточних процесів, але й навчання команди, підвищення її кваліфікації та адаптацію до нових викликів. Регулярні тренінги та воркшопи можуть бути корисними для покращення навичок команди та впровадження нових методів роботи.

У підсумку, моніторинг в Agile-проектах є ключовим елементом успішного управління проектами. Використання структурованого підходу до моніторингу, який включає регулярне відстеження прогресу, проведення ретроспектив та постійне вдосконалення процесів, дозволяє забезпечити високу ефективність та якість розробки програмного забезпечення. Інтеграція сучасних інструментів та технологій для збору, аналізу та візуалізації даних сприяє більш точному та своєчасному прийняттю рішень. Врахування зворотного зв'язку від усіх зацікавлених сторін забезпечує відповідність кінцевого продукту їхнім очікуванням та потребам. Таким чином, ефективний моніторинг сприяє підвищенню гнучкості, прозорості та загальної продуктивності команди, що є критично важливим для успішного виконання проектів у сучасних умовах.

1.3 Дослідження методів процес-майнінг в задачах мониторингу Agile проектів

Процес майнінг (Process Mining) є ефективним методом інтелектуального аналізу процесів, який дозволяє автоматично виявляти, контролювати та покращувати реальні процеси, використовуючи дані з інформаційних систем. Цей підхід базується на аналізі журналів подій (event logs) і дозволяє отримувати візуальне представлення процесів у вигляді графів.

Основні етапи процес-майнінгу:

Збір даних (Event Log):

– подія: дія, виконана в рамках процесу (наприклад, коміт, створення pull request);

– атрибути події: ідентифікатор події, час виконання, користувач, тип події, інші релевантні дані.

Побудова моделі процесу:

– використання методів процес-майнінгу для створення моделі процесу на основі зібраних даних.

Формула для розрахунку ймовірності переходу між подіями:

$$P(a \to b) = \frac{\text{Кількістьпереходіввід}a\text{до}b}{\text{Загальнакількістьподій}a} \quad (1.1)$$

де $P(a \to b)$ – ймовірність переходу від події a до події b.

Аналіз та візуалізація процесу:

– створення графового представлення процесу, де вузли представляють події, а ребра – переходи між подіями;

– використання метрик для оцінки ефективності процесу, таких як середній час виконання, затримки, частота виконання певних дій.

Порівняння з еталонною моделлю:

– порівняння отриманої моделі процесу з еталонною моделлю для виявлення відхилень та потенційних проблем.

Формула для розрахунку відхилення:

$$D\left(P_{real}, P_{planned}\right) = \sum_{i=1}^{n} \left| P_{real}(a_i \to b_i) - P_{planned}(a_i \to b_i) \right| \quad (1.2)$$

де $D(P_{real}, P_{planned})$ – сумарне відхилення між реальним процесом P_{real} та запланованим процесом $P_{planned}$;

$a_i \rightarrow b_i$ – перехід від події a_i до події b_i.

Приклади методів процес-майнінгу:

– AlphaMiner: використовується для виявлення послідовностей подій та побудови початкових моделей процесів;

– Heuristic Miner: використовується для побудови моделей процесів з урахуванням частоти виконання подій;

– Inductive Miner: побудова моделей процесів на основі індуктивних правил. Використовує розподіл подій та підпроцесів для побудови комплексних моделей;

– Fuzzy Miner: використовується для аналізу складних процесів з великою кількістю подій;

– Process Mining using ProM: використання програмного інструменту ProM для аналізу та візуалізації процесів. Підтримує широкий спектр алгоритмів для процес-майнінгу, включаючи AlphaMiner, Heuristic Miner, Inductive Miner та інші.

Використання обробленої інформації про події процесу Agile розробки дозволяє ефективно порівнювати реальний процес розробки із запланованим. Це досягається шляхом створення графового представлення фактичного процесу, що забезпечує прозорість та точність моніторингу. Методи процес-майнінгу, такі як AlphaMiner, Heuristic Miner, Inductive Miner, Fuzzy Miner та ProM, дозволяють виявляти та усувати відхилення, покращуючи загальну ефективність процесів розробки.

Процес-майнінг є потужним інструментом для аналізу та вдосконалення процесів у проектах, зокрема в рамках Agile методологій. Нижче представлено таблицю з основними методами процес-майнінгу, які можуть бути

використані для моніторингу та аналізу Agile проектів. Перелік цих методів перелічені в таблиці 1.1.

Таблиця 1.3 - Методи процес-майнінгу

Метод процес-майнінгу	Опис	Переваги	Недоліки
Alpha Miner	Метод, що використовує алгоритми для виявлення структурних моделей процесів на основі журналів подій.	Чітке виявлення послідовності дій; Відсутність необхідності попередніх припущень.	Може створювати надмірно складні моделі; Чутливий до шуму у даних.
Heuristic Miner	Метод, що використовує евристики для виявлення частотних залежностей у процесах.	Ефективний для складних та великих даних; Стійкий до шуму.	Потребує налаштування евристик; Може не виявляти рідкісні, але важливі залежності.
Inductive Miner	Метод, що створює узагальнені моделі процесів з високою точністю.	Висока точність моделей; Підтримка узагальнених процесів.	- Високі вимоги до обчислювальних ресурсів.

Продовження таблиці 1.3

Метод процес-майнінгу	Опис	Переваги	Недоліки
Fuzzy Miner	Метод, що дозволяє візуалізувати складні та нечіткі процеси.	Добре підходить для візуалізації; Підтримка нечітких та неоднозначних даних.	Може створювати складні для інтерпретації моделі; Потребує налаштування параметрів.
Process Mining using ProM	Інструмент, що об'єднує різні алгоритми процес-майнінгу для аналізу журналів подій.	Великий набір інструментів та алгоритмів. - Підтримка різних форматів даних.	Складність у використанні для новачків; Високі вимоги до даних та їхньої підготовки.

Alpha Miner є одним із найбільш базових методів процес-майнінгу, який використовує алгоритми для виявлення структурних моделей процесів на основі журналів подій. Цей метод аналізує послідовність дій і намагається створити чітку модель процесу, що відображає порядок виконання завдань. Alpha Miner виділяється тим, що чітко визначає послідовність дій, що є його основною перевагою. Цей метод не вимагає попередніх припущень щодо структури процесу, що дозволяє йому бути більш гнучким та універсальним. Однак, Alpha Miner має деякі недоліки. Наприклад, він може створювати надмірно складні моделі, що ускладнює їх інтерпретацію та використання. Крім того, Alpha Miner є чутливим до шуму у даних, що може призвести до викривлення результатів та побудови некоректних моделей. Незважаючи на ці

обмеження, Alpha Miner залишається популярним методом через свою простоту та здатність швидко надавати корисні результати.

Heuristic Miner використовує евристики для виявлення частотних залежностей у процесах. Цей метод дозволяє ефективно працювати зі складними та великими даними, виявляючи найбільш важливі зв'язки між подіями. Однією з основних переваг Heuristic Miner є його здатність справлятися зі складними та великими даними, що робить його ефективним інструментом для аналізу великих бізнес-процесів. Він стійкий до шуму, що дозволяє отримувати точні результати навіть у випадках, коли дані містять помилки або неповні записи. Проте, Heuristic Miner має деякі недоліки. Наприклад, він потребує налаштування евристик, що може бути складним завданням для користувачів без спеціальної підготовки. Крім того, цей метод може не виявляти рідкісні, але важливі залежності між подіями, що може призвести до упущення критично важливої інформації.

Inductive Miner створює узагальнені моделі процесів з високою точністю. Цей метод дозволяє будувати моделі, які легко масштабуються та підтримують узагальнені процеси. Однією з ключових переваг Inductive Miner є висока точність створюваних моделей, що забезпечує більш глибоке розуміння процесів та їхньої структури. Цей метод підтримує узагальнені процеси, що дозволяє використовувати його для аналізу різних типів бізнес-процесів. Однак, Inductive Miner має високі вимоги до обчислювальних ресурсів, що може бути проблемою при аналізі дуже великих обсягів даних або складних процесів. Незважаючи на це, Inductive Miner залишається потужним інструментом для аналізу та оптимізації бізнес-процесів.

Fuzzy Miner призначений для візуалізації складних та нечітких процесів. Цей метод добре підходить для виявлення та відображення неоднозначних даних і складних взаємозв'язків у процесах. Однією з основних переваг Fuzzy Miner є його здатність візуалізувати складні та нечіткі процеси, що робить його корисним інструментом для аналізу процесів, які важко формалізувати. Fuzzy

Miner підтримує роботу з нечіткими та неоднозначними даними, що дозволяє використовувати його у випадках, коли дані мають високий рівень невизначеності. Проте, цей метод може створювати складні для інтерпретації моделі, що ускладнює їх використання для прийняття рішень. Крім того, Fuzzy Miner потребує налаштування параметрів, що може бути складним завданням для новачків.

ProM є потужним інструментом для процес-майнінгу, що об'єднує різні алгоритми та методи для аналізу журналів подій. ProM підтримує різні формати даних і пропонує широкий спектр інструментів для аналізу. Однією з основних переваг ProM є його великий набір інструментів та алгоритмів, що дозволяє проводити комплексний аналіз бізнес-процесів. ProM підтримує різні формати даних, що робить його універсальним інструментом для аналізу процесів у різних середовищах. Проте, ProM може бути складним у використанні для новачків через велику кількість функцій та налаштувань. Крім того, цей інструмент має високі вимоги до даних та їхньої підготовки, що може бути додатковим викликом для користувачів.

Методи процес-майнінгу є важливими інструментами для моніторингу та аналізу Agile проектів. Вони дозволяють виявляти та візуалізувати процеси, аналізувати їхню ефективність та виявляти можливі вузькі місця. Використання процес-майнінгу допомагає підвищити прозорість процесів, забезпечити якість виконання завдань та постійно вдосконалювати методологію роботи команди. Процес-майнінг дозволяє отримати об'єктивну картину про те, як виконуються завдання, що забезпечує більш точне планування та прогнозування майбутніх спринтів.

Завдяки використанню методів процес-майнінгу, команди можуть виявляти проблеми на ранніх стадіях та вживати заходів для їх усунення. Це сприяє зниженню ризиків та підвищенню загальної ефективності проекту. Крім того, процес-майнінг дозволяє аналізувати історичні дані та виявляти

тренди, що може бути корисним для довгострокового планування та стратегічного управління проектами.

Використання різних методів процес-майнінгу, таких як Alpha Miner, Heuristic Miner, Inductive Miner, Fuzzy Miner та ProM, дозволяє здійснювати комплексний аналіз процесів, враховуючи всі аспекти їх виконання. Кожен із цих методів має свої переваги та недоліки, що дозволяє обирати найбільш підходящий метод залежно від конкретних потреб проекту. Використання процес-майнінгу у поєднанні з іншими методами аналізу та управління проектами дозволяє забезпечити більш комплексний та ефективний підхід до моніторингу та вдосконалення процесів розробки програмного забезпечення.

1.4 Постановка задачі дослідження

Моніторинг зазвичай виконується на етапі ретроспективи. В Agile методологіях, ретроспектива є ключовим етапом, на якому команда аналізує виконану роботу за спринт, обговорює досягнення та проблеми, а також визначає шляхи вдосконалення процесів. Моніторинг у цей період дозволяє отримати зворотний зв'язок та зробити висновки щодо ефективності виконання завдань. Проте, більшість даних, зібраних під час ретроспективи, є суб'єктивними, що створює певні складнощі у покращенні проектів.

Ці дані є суб'єктивними, тому виникають складності в покращенні проектів. Суб'єктивність даних, отриманих під час ретроспектив, обумовлена особистими враженнями та оцінками членів команди. Це може призвести до нерівномірного розподілу уваги до проблем та до упущення важливих аспектів процесу розробки. Така ситуація ускладнює об'єктивний аналіз та ухвалення ефективних рішень щодо вдосконалення процесів.

Для об’єктивного аналізу та поліпшення на етапі ретроспективи необхідно побудувати модель послідовності робіт проекту з використанням методів процес-майнінгу. Для досягнення об'єктивності на етапі

ретроспективи необхідно використовувати методи процес-майнінгу. Це дозволяє створити точну модель послідовності робіт проекту на основі реальних даних, що забезпечує глибший аналіз процесів та виявлення критичних точок для вдосконалення.

Першим кроком для створення об'єктивної моделі є формування журналу подій. Цей журнал повинен містити всі ключові події проекту, такі як створення завдань, коміти, pull requests, code reviews та інші важливі дії. Інформація має бути зібрана з систем управління версіями, таких як GitHub, у формі, придатній для подальшого аналізу методами процес-майнінгу. Журнал подій дозволяє відстежувати кожен крок у процесі розробки, забезпечуючи точність та повноту даних.

Наступним етапом є виявлення ключових залежностей у процесі розробки. Ці залежності визначають, як різні події взаємодіють одна з одною та як вони впливають на загальний процес. Темпоральна упорядкованість подій, тобто їхній хронологічний порядок, є важливим аспектом для розуміння взаємозв'язків між діями.

Використання методу процес-майнінгу для моніторингу Agile проектів дозволяє мінімізувати недоліки, пов'язані з суб'єктивністю даних та можливістю виникнення хаосу. Цей метод забезпечує структурованість, прозорість та об'єктивність у процесі управління проектами, що сприяє підвищенню ефективності та якості розробки програмного забезпечення.

Об'єктом дослідження є процеси моніторингу проектів гнучкої розробки програмного забезпечення. Та предметом дослідження – методи моніторингу процесів Agile розробки з використанням інструментів процес-майнінгу.

Таким чином, об'єктом дослідження є процеси моніторингу Agile проектів, а предметом дослідження є методи, які дозволяють здійснювати цей моніторинг з максимальною точністю та ефективністю, зокрема за допомогою процес-майнінгу.

2 АНАЛІЗ ПРОЦЕСНИХ МЕТОДІВ МОНІТОРИНГУ AGILE-ПРОЕКТІВ

2.1 Розробка підходу до моніторингу Agile-проектів з використанням інструментарію інтелектуального аналізу процесів

Моніторинг Agile проектів є важливим елементом успішного управління, оскільки він дозволяє контролювати прогрес, виявляти проблеми на ранніх стадіях та швидко реагувати на зміни. Інтелектуальний аналіз процесів забезпечує автоматизацію та підвищену точність моніторингу, що є критичним для підтримання ефективності та якості проектів у динамічному середовищі.

Основний підхід до моніторингу Agile проектів включає декілька ключових етапів. Визначення цілей та метрик:

– на початковому етапі важливо чітко визначити цілі моніторингу. Ці цілі можуть включати забезпечення своєчасного виконання завдань, підвищення якості продукту, зменшення кількості дефектів та інші;

– вибір відповідних метрик для оцінки ефективності процесів. Метрики повинні бути зрозумілими, вимірюваними та релевантними для всіх учасників проекту. Основні метрики можуть включати Cycle Time (CT), Lead Time (LT), Time in Status (In Progress) та Time in Status (Review).

Збір даних:

– використання автоматизованих інструментів для збору даних з різних джерел, таких як системи управління версіями (GitHub), системи управління завданнями (JIRA, Trello) та інші;

– дані повинні бути структуровані у вигляді журналу подій, що містить інформацію про час виконання, статуси завдань та відповідальних осіб. Це

дозволяє забезпечити цілісність та точність зібраних даних для подальшого аналізу.

Аналіз даних:

– застосування методів процес-майнінгу для аналізу зібраних даних. Це включає виявлення послідовностей подій, виявлення залежностей та оцінку ефективності процесів;

– використання таких методів, як AlphaMiner, Heuristic Miner, Inductive Miner, Fuzzy Miner та Process Mining using ProM, для глибокого аналізу.

Зворотний зв'язок та коригувальні дії:

– на основі аналізу та візуалізації даних, команда може приймати коригувальні дії для покращення процесу розробки;

– регулярні ретроспективи та огляди результатів допомагають підтримувати постійне вдосконалення процесів. Це включає аналіз успіхів і невдач, визначення причин проблем та розробку плану дій для їх усунення.

Метрики є ключовим елементом моніторингу, оскільки вони дозволяють кількісно оцінити ефективність процесів та виявити потенційні проблеми. Основні метрики, що використовуються у моніторингу Agile проектів, включають:

– Cycle Time (CT): Час, необхідний для виконання конкретного завдання від моменту його початку до завершення;

– Lead Time (LT): Загальний час від моменту створення завдання до його завершення;

– Time in Status (In Progress): Час, протягом якого завдання знаходиться у статусі "В роботі";

– Time in Status (Review): Час, протягом якого завдання знаходиться у статусі "Перевірка".

Таблиця 2.1 – Таблиця метрик

Метрика	Опис
Cycle Time (CT)	Час, необхідний для виконання завдання від початку до кінця
Lead Time (LT)	Загальний час від створення завдання до його завершення
Time in Status (In Progress)	Час, протягом якого завдання знаходиться у статусі "В роботі"
Time in Status (Review)	Час, протягом якого завдання знаходиться у статусі "Перевірка"

Оновлена схема моніторингу складається з чотирьох етапів, які забезпечують структурований підхід до моніторингу та вдосконалення процесів. Цей метод зображений на рисунку 2.1.

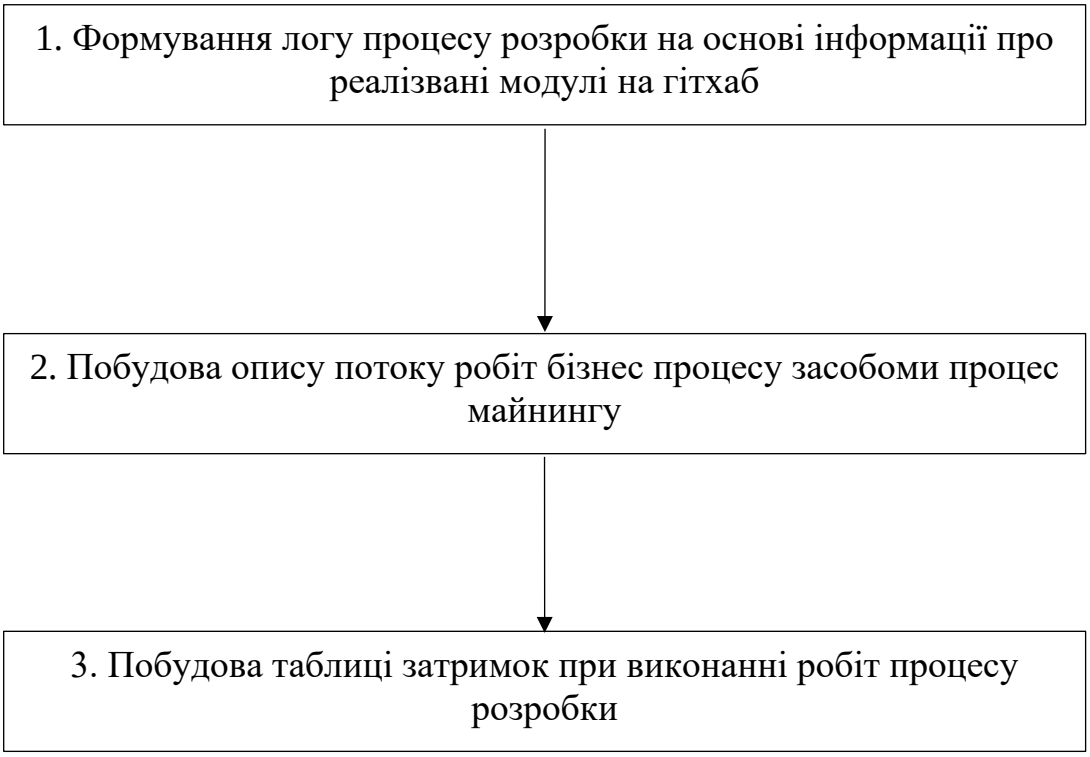


Рисунок 2.1– Оновлена схема моніторингу

Побудова моделі (аналіз User Story): На цьому етапі аналізуються User Stories для побудови моделі процесу. Це допомагає зрозуміти поточні процеси та виявити можливі проблеми на ранніх етапах.

Управління з використанням процесної моделі (Спринти у Scrum): Відбувається управління процесом через спринти. Кожен спринт базується на процесній моделі, що дозволяє підтримувати структурованість і зменшити хаос через постійні зміни.

Моніторинг виконання (Збір та аналіз інформації з GitHub): Збір та аналіз даних з GitHub дозволяють отримати об'єктивну інформацію про прогрес проекту. Це забезпечує прозорість та точність даних, що допомагає уникнути проблем з документацією.

Вдосконалення процесної моделі (Аналіз отриманих даних на Retrospective): Аналіз даних, зібраних під час ретроспектив, дозволяє вдосконалювати процеси на основі реальних результатів. Це сприяє постійному покращенню та адаптації процесів до нових вимог.

Переваги методу процес-майнингу:

– структурованість: використання процесної моделі дозволяє зберегти структурованість та уникнути хаосу;

– прозорість: збір об'єктивних даних з GitHub забезпечує прозорість процесів та точність у моніторингу прогресу;

– постійне вдосконалення: аналіз даних на ретроспективах сприяє постійному покращенню процесів, що підвищує загальну ефективність команди.

Метод процес-майнингу в рамках гнучкого підходу до моніторингу дозволяє ефективно керувати проектами, мінімізуючи ризики, пов'язані з постійними змінами та складністю в забезпеченні прозорості. Використання чітко визначених етапів моніторингу забезпечує структурованість, прозорість та постійне вдосконалення процесів, що є критично важливим для успішного завершення проектів у сучасному динамічному середовищі IT.

2.2 Розробка методу моніторингу процесу розробки Agile-проектів з використанням інтелектуального аналізу процесів

Моніторинг процесу розробки Agile-проектів з використанням інтелектуального аналізу процесів забезпечує глибокий аналіз і візуалізацію виконуваних процесів. Цей підхід дозволяє виявити послідовність дій, виявити відхилення від запланованого процесу, оцінити продуктивність команди та оптимізувати процеси для досягнення кращих результатів.

Для ефективного моніторингу Agile проектів розроблено процесний метод, який включає чотири основні етапи:

– етап 1. формування логу процесу розробки на основі інформації про реалізвані модулі на Github**;**

– етап 2. побудова опису потоку робіт бізнес-процесу засобами процес майнингу**;**

– етап 3. побудова таблиці затримок при виконанні робіт процесу розробки**.**

Моніторинг Agile проектів потребує систематичного аналізу даних для виявлення вузьких місць та оптимізації процесів. Використання метрик дозволяє кількісно оцінити ефективність команди та швидкість виконання завдань.

Для того щоб забезпечити ефективний моніторинг Agile проектів, необхідно створити загальну схему послідовності етапів моніторингу. Смарт карта цього процесу може виглядати, як на рисунку 2.1.

Рисунок 2.1 – Смарт карта процесу моніторингу

Для оцінки ефективності процесів використовуються метрики та відповідні формули для їх розрахунку.

Cycle Time (CT): Час, необхідний для завершення завдання від початку до кінця.

$$CT = T_{completed} - T_{started} \quad (4.1)$$

де CT – сумарний час виконання завдання;

$T_{completed}$ – дата завершення завдання;

$T_{started}$ – дата початку виконання завдання.

Lead Time (LT): Час, який проходить від моменту, коли завдання з'являється у беклозі, до моменту його завершення.

$$LT = T_{completed} - T_{created} \quad (4.2)$$

де LT – сумарний час від створення до завершення завдання;

$T_{completed}$ – дата завершення завдання;

$T_{created}$ – дата створення завдання.

Time in Status (TiS): Час, протягом якого завдання перебуває в певному статусі (наприклад, "In Progress", "Review").

$$TiS = T_{status_{exit}} - T_{status_{entry}} \quad (4.3)$$

де TiS – час, який завдання перебуває у певному статусі;

$T_{status_{exit}}$ – час, коли статус завдання було змінено на інший;

$T_{status_{entry}}$ – час, коли статус завдання було змінено на поточний.

Використання цих метрик дозволяє об'єктивно оцінити продуктивність команди та виявити потенційні проблеми у процесі розробки. Це, в свою чергу, сприяє постійному вдосконаленню процесів та підвищенню ефективності роботи команди.

Alpha алгоритм є одним з основних методів процес-майнінгу, що використовується для виявлення процесів з журналів подій (event logs). Основна мета Alpha алгоритму – виявити послідовність подій та побудувати модель процесу. Основні кроки Alpha алгоритму показані на рисунку 2.2.

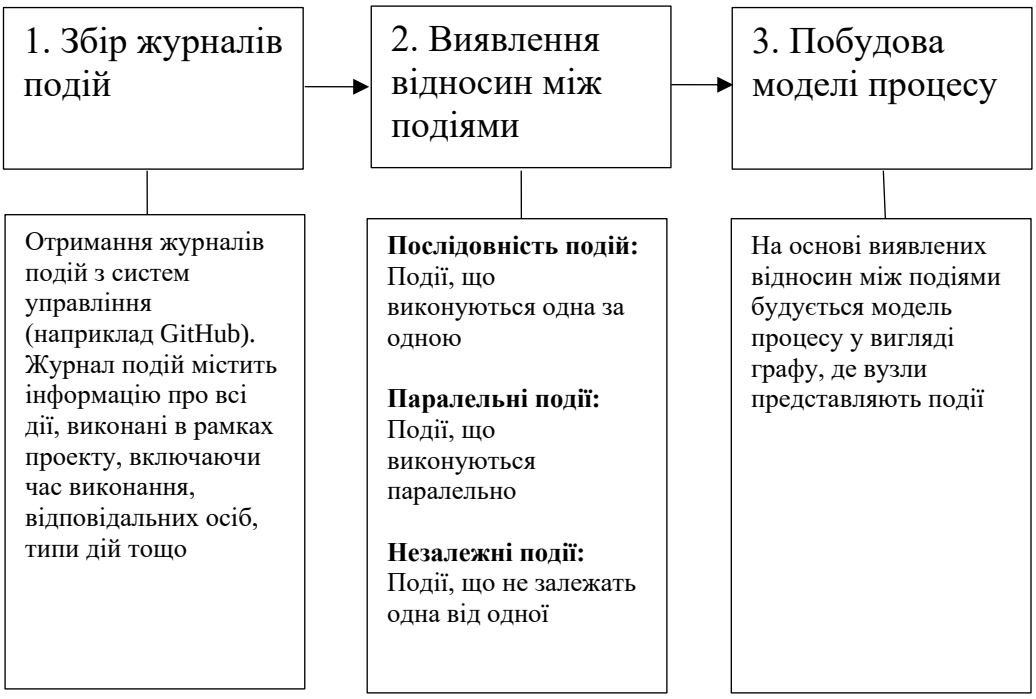


Рисунок 2.2 – Основні кроки Alpha алгоритму

Задачі, що виконуються паралельно, послідовно та незалежно:

– послідовні задачі: задачі, що виконуються одна за одною. Наприклад, після завершення розробки (In Progress) виконуються Code Review та, нарешті, злиття коду (Merge);

– паралельні задачі: задачі, що виконуються одночасно різними членами команди. Наприклад, два фронтенд-розробники можуть одночасно працювати над різними фічами проекту**;**

– незалежні задачі: задачі, що не залежать одна від одної та можуть виконуватися незалежно. Наприклад, розробка фронтенду та бекенду можуть виконуватися незалежно, доки не буде необхідності інтеграції.

3 РОЗРОБКА ПРОЕКТУ AGILE МОНИТОРИНГУ

Проект спрямований на вдосконалення процесу моніторингу Agile проектів з використанням інструментів процес-майнінгу. Основна ідея полягає у створенні автоматизованої системи, яка буде збирати дані з GitHub, аналізувати їх і формувати темпоральні та ключові залежності для покращення управління проектами. Для реалізації цього проекту використовується скрипт, написаний на Python, який автоматично збирає та обробляє дані з GitHub та аналізує журнал GitHub за допомогою Alpha алгоритму.

3.1 Формулювання основи (статуту) проекту

В сучасному світі IT-індустрія швидко розвивається, що вимагає від компаній гнучкості та ефективності у процесах розробки програмного забезпечення. Впровадження Agile методологій дозволяє значно покращити ці процеси, забезпечуючи адаптивність та швидкість реагування на зміни. Проте, для досягнення максимальних результатів, важливо мати ефективний механізм моніторингу та аналізу виконання завдань. Моніторинг Agile проектів з використанням інструментарію інтелектуального аналізу процесів дозволяє:

– виявляти вузькі місця у процесах та своєчасно їх усувати**;**

– забезпечувати прозорість виконання задач та підвищувати відповідальність членів команди**;**

– проводити об'єктивний аналіз на основі реальних даних, що дозволяє приймати обґрунтовані рішення**;**

– підвищувати загальну ефективність та якість розробки програмного забезпечення**.**

Цей проект є надзвичайно актуальним для забезпечення конкурентоспроможності та інноваційності компаній у таких сферах, як розробка програмного забезпечення.

Таблиця 3.1 – Базовий план проекту дослідження

Основне завдання	Опис завдання
Дослідження процесних моніторингу Agile-проектів (розділ 1 дослідження)	Перший крок полягає у детальному аналізі поточних методів моніторингу Agile-проектів. Це включає дослідження та аналіз поточних процесів, використовуваних інструментів, а також визначення можливих проблем і обмежень існуючих методів
Визначення недоліків методів моніторингу Agile-проектів (розділ 2 дослідження)	Необхідно визначити недоліків методів, зокрема, способи які дозволять подолати ці недоліки
Розробка методу моніторингу інструментами процес-майнингу (розділ 3 дослідження)	Створення детального плану проекту розробки методу моніторингу, що включає етапи впровадження, критерії успіху, очікувані ризики, та стратегії їх мінімізації
Експериментальна перевірка (розділ 4 проекту)	Застосування методу моніторингу Agile-проектів з використанням процес-майнингу. Це дозволить оцінити потенційні покращення та визначити необхідні корективи в системі

Продовження таблиці 3.1

Основне завдання	Опис завдання
Моніторинг виконання проекту (розділ 5 проекту)	Постійний моніторинг прогресу виконання проекту з використанням створеної моделі. Виявлення та усунення відхилень від плану, забезпечення відповідності завдань запланованим показникам.
Аналіз та вдосконалення процесної моделі (розділ 6 проекту)	Після завершення кожного циклу розробки (спринту) проводиться ретроспектива для аналізу отриманих результатів. На основі зібраних даних вносяться корективи у процесну модель для її вдосконалення.
Впровадження методу в процес розробки програмного забезпечення (не буде розглянуто в дослідженні)	Впровадження та застосування проекту в процес розробки з ціллю модифікації та покращення показників останньої. Цей етап включає в себе подальшу підтримку результатів проекту

Таблиця 3.2 – Команда проекту

Роль	Обов'язки
Project Manager	Відповідальний за загальне управління проектом, координація дій команди, вирішення конфліктів, прийняття стратегічних рішень, контроль виконання проектних завдань
Junior Python Developer	Розробка коду, аналіз даних, побудова моделі процесу, пропозиції щодо покращення процесу, участь у прийнятті рішень

Продовження таблиці 3.2

Роль	Обов'язки
Middle Python Developer	Розробка коду, аналіз даних, побудова моделі процесу, пропозиції щодо покращення процесу, участь у прийнятті рішень

Проект розрахований на 4 ітерації, кожна тривалістю 2-3 тижні. Загальна тривалість проекту – 10 тижнів.

Часта заміна кадрів на проекті не передбачена. Правильним кандидатом на позицію розробника є розробник, що вже працював зі схожими технологіями та програмними мовами. Передбачено стандартний робочий графік – 8 годин по п’ять днів на тиждень.

3.2 Основні етапи проекту

Проект моніторингу Agile-проектів з використанням інструментів процес-майнінгу охоплює кілька ключових етапів, які забезпечують повний життєвий цикл від початкового дослідження до кінцевого впровадження. Цей проект передбачає систематичний підхід до аналізу та оптимізації процесів розробки, що дозволяє отримати глибоке розуміння виконуваних процесів та забезпечити їхню прозорість.

Процес моніторингу складається з чотирьох основних ітерацій, кожна з яких має чітко визначені цілі, конкретні задачі та очікувані результати. На кожному етапі виконуються певні дії, спрямовані на досягнення ефективного моніторингу та аналізу процесів розробки. Смарт карта цих ітерацій зображена на рисунку 3.1.

Перший етап, дослідження, включає збір необхідних даних та аналіз поточного стану процесів. Це забезпечує основу для подальшого розроблення та впровадження інструментів процес-майнінгу. Другий етап, розробка,

передбачає створення моделей процесів та розробку інструментів для моніторингу. Третій етап, аналіз, фокусується на детальному вивченні зібраних даних, оцінці ефективності процесів та виявленні можливих відхилень. Четвертий етап, впровадження, передбачає інтеграцію розроблених інструментів у реальне середовище, проведення навчання команди та постійне вдосконалення процесів на основі отриманих даних.

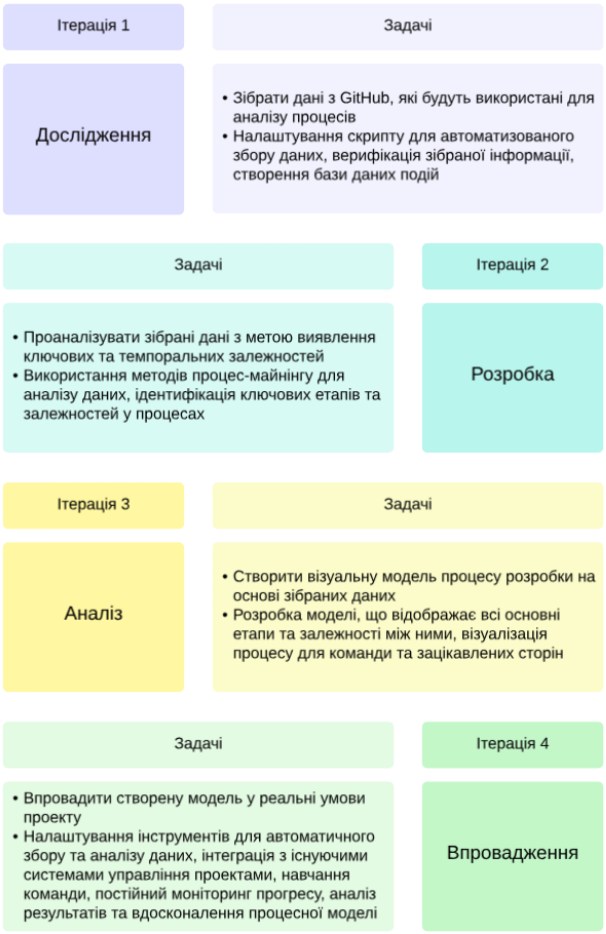


Рисунок 3.1 – Ітерації проекту

Кожна ітерація проекту має свої специфічні завдання та цілі, що дозволяє поетапно підходити до вирішення проблем та вдосконалення процесів розробки. Такий підхід забезпечує високу гнучкість, можливість швидкої адаптації до змін та постійне покращення ефективності роботи

команди. Очікувані результати кожної ітерації дозволяють отримати чітке уявлення про прогрес проекту та досягнення поставлених цілей.

Таким чином, проект моніторингу Agile проектів з використанням інструментів процес-майнінгу є комплексним процесом, що включає кілька ключових етапів та ітерацій, кожна з яких має свої цілі, задачі та очікувані результати. Цей підхід забезпечує повний життєвий цикл проекту від дослідження до впровадження, сприяючи досягненню високої ефективності та якості процесів розробки.

Життєвий цикл інформаційної системи забезпечує послідовність дій від планування до завершення проекту, що сприяє ефективному управлінню та успішному впровадженню нових систем в організацію.

Він охоплює всі етапи від концептуального планування до завершення та заміни системи. Розглянемо кожен етап детальніше:

На цьому етапі визначаються потреби організації в новій інформаційній системі. Важливо чітко сформулювати проблеми, які повинні бути вирішені, та можливості, що надаються новою системою. Етап ініціації завершується складанням статуту проекту, який обґрунтовує необхідність системи та визначає її основні цілі та завдання.

Етап планування включає деталізацію всіх аспектів проекту. Це включає визначення обсягу робіт, розробку графіку виконання завдань, оцінку необхідних ресурсів та розробку плану управління ризиками. На цьому етапі складаються детальні плани, які використовуватимуться для керування проектом на всіх наступних етапах.

Під час етапу розробки створюється сама інформаційна система. Розробляються її архітектура, програмний код, інтерфейси користувача та інтеграція з іншими системами. Цей етап завершується створенням технічної документації, що описує функціонування системи, та тестових сценаріїв для перевірки її працездатності.

На етапі впровадження система інтегрується в існуюче середовище організації. Встановлюється програмне забезпечення, налаштовується система, проводиться навчання користувачів та тестування системи в реальних умовах. Всі дії документуються у вигляді настанов для користувачів та звітів про тестування.

Після впровадження починається етап експлуатації, протягом якого забезпечується безперебійне функціонування системи. Моніторинг роботи системи дозволяє вчасно виявляти та виправляти помилки, а також здійснювати оновлення та модернізацію системи для підтримання її актуальності.

Останній етап життєвого циклу системи включає офіційне завершення проекту та оцінку його результатів. Проводиться аналіз досягнутих результатів, складаються фінальні звіти та документи про прийняття системи. Оцінюються досягнення та невдачі проекту, готується план на випадок завершення експлуатації системи або її заміни новою.

3.3 Розробка детального плану проекту

Враховуючи команду, обов’язки та мету проекту, було розроблено наступний план на 4 ітерації. Цей план включає в себе детальне планування всіх етапів проекту. План проекту зображений на рисунку 3.2.

	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		**Вдосконалення процесу моніторингу Agile проектів**	**29 days**	**Mon 04.03.24**	**Thu 11.04.24**		
2		**Ітерація 1: Дослідження**	**12 days**	**Mon 04.03.24**	**Tue 19.03.24**		
3		Зібрати дані з GitHub	2 days	Mon 04.03.24	Tue 05.03.24		Junior Python Developer
4		Форматування даних у структуровані журнали подій	5 days	Wed 06.03.24	Tue 12.03.24	3	Junior Python Developer
5		Розробка та налаштування скриптів	3 days	Mon 04.03.24	Wed 06.03.24		Middle Python Developer
6		Верифікація зібраної інформації	2 days	Wed 13.03.24	Thu 14.03.24	4	Project manager
7		Створення бази даних подій для подальшого аналізу	3 days	Fri 15.03.24	Tue 19.03.24	6	Middle Python Developer
8		**Ітерація 2: Розробка**	**15 days**	**Thu 07.03.24**	**Wed 27.03.24**		
9		Виконати попередній аналіз зібраних даних для виявлення ключових залежностей	2 days	Wed 20.03.24	Thu 21.03.24	7	Middle Python Developer
10		Використання методів процес-майнінгу для ідентифікації темпоральних залежностей у процесах	2 days	Fri 22.03.24	Mon 25.03.24	9	Middle Python Developer
11		Застосування Alpha алгоритму та інших методів процес-майнінгу для створення початкових моделей процесів	3 days	Thu 07.03.24	Mon 11.03.24	5	Middle Python Developer
12		Визначення ключових етапів у процесах розробки та їх взаємозв'язків	2 days	Tue 26.03.24	Wed 27.03.24	10	Middle Python Developer
13		Візуалізація процесів	3 days	Tue 12.03.24	Thu 14.03.24	11	Middle Python Developer
14		**Ітерація 3: Аналіз**	**14 days**	**Fri 15.03.24**	**Wed 03.04.24**		
15		Побудова візуальної моделі процесу розробки	3 days	Fri 15.03.24	Tue 19.03.24	13	Project manager
16		Використання інструментів візуалізації для створення графів та діаграм	2 days	Wed 20.03.24	Thu 21.03.24	15	Junior Python Developer
17		Розробка моделі процесу	2 days	Thu 28.03.24	Fri 29.03.24	12	Middle Python Developer
18		Візуалізація процесу для команди та зацікавлених сторін	2 days	Mon 01.04.24	Tue 02.04.24	16;17	Junior Python Developer
19		Використання отриманої моделі для планування та оптимізації	1 day	Wed 03.04.24	Wed 03.04.24	18	Middle Python Developer
14		**Ітерація 3: Аналіз**	**14 days**	**Fri 15.03.24**	**Wed 03.04.24**		
15		Побудова візуальної моделі процесу розробки	3 days	Fri 15.03.24	Tue 19.03.24	13	Project manager
16		Використання інструментів візуалізації для створення графів та діаграм	2 days	Wed 20.03.24	Thu 21.03.24	15	Junior Python Developer
17		Розробка моделі процесу	2 days	Thu 28.03.24	Fri 29.03.24	12	Middle Python Developer
18		Візуалізація процесу для команди та зацікавлених сторін	2 days	Mon 01.04.24	Tue 02.04.24	16;17	Junior Python Developer
19		Використання отриманої моделі для планування та оптимізації	1 day	Wed 03.04.24	Wed 03.04.24	18	Middle Python Developer
20		**Ітерація 4: Впровадження**	**11 days**	**Thu 28.03.24**	**Thu 11.04.24**		
21		Інтеграція створеної моделі процесу у реальні умови проекту	2 days	Thu 28.03.24	Fri 29.03.24	8	Project manager;Junior Python Developer
22		Налаштування системи для автоматичного збору та аналізу даних	3 days	Mon 01.04.24	Wed 03.04.24	21	Project manager
23		Налаштування та впровадження інструментів для автоматичного збору та аналізу даних	2 days	Mon 01.04.24	Tue 02.04.24	21	Middle Python Developer
24		Інтеграція з існуючими системами управління проектами	2 days	Thu 04.04.24	Fri 05.04.24	22;23	Middle Python Developer
25		Навчання команди використанню нових інструментів та методів	2 days	Mon 08.04.24	Tue 09.04.24	24	Middle Python Developer
26		Аналіз результатів та виявлення відхилень	1 day	Mon 08.04.24	Mon 08.04.24	24	Project manager
27		Вдосконалення процесної моделі на основі отриманих даних та зворотного зв'язку від команди	3 days	Tue 09.04.24	Thu 11.04.24	26	Project manager

Рисунок 3.2 – Тестовий план розробки

Загальний план робіт має протяжність в 29 днів. Найдовша ітерація займає 15 днів. А вартість проекту 2808,00$ (дві тисячі вісімсот вісім).

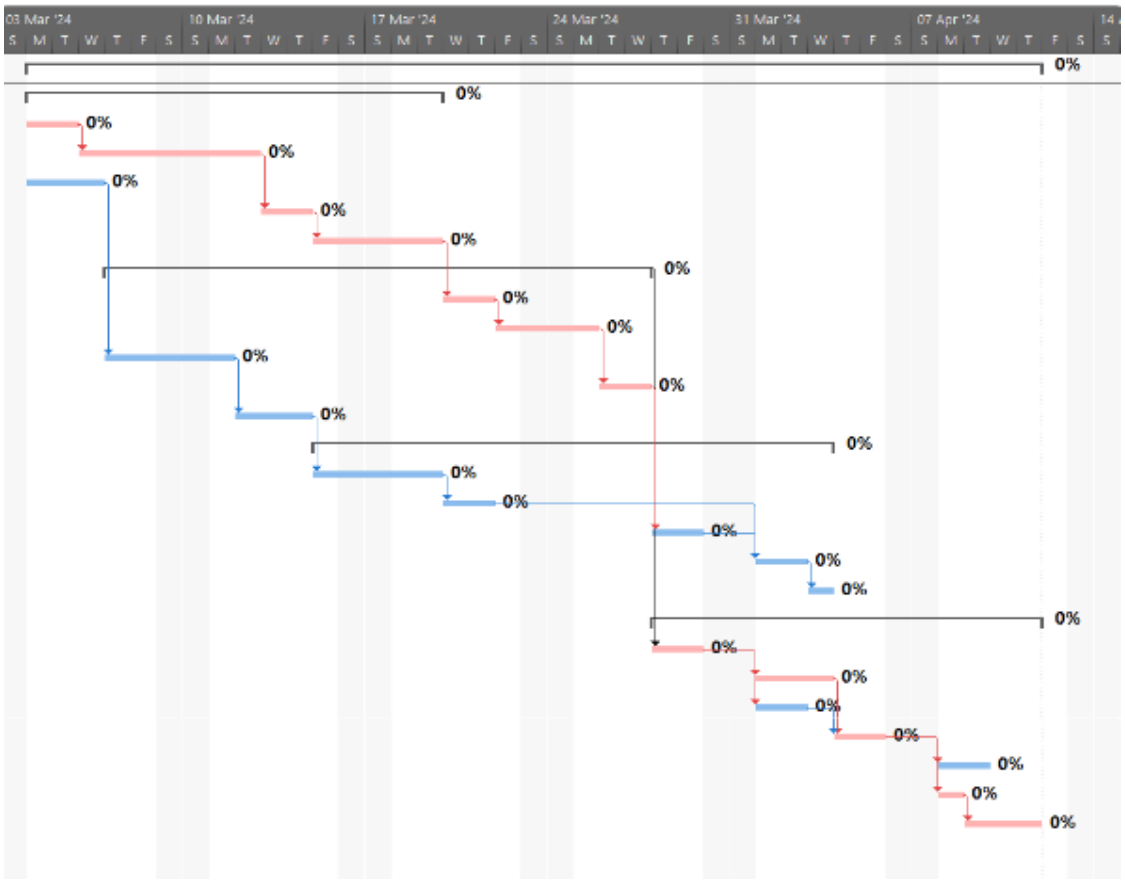


Рисунок 3.3 – Діаграма Ганта проекту

3.4 Висновки з планування проекта

Планування проекту моніторингу Agile-проектів з використанням інструментів процес-майнінгу показало, що систематичний підхід до розробки та впровадження процесних моделей може значно покращити якість та ефективність роботи команди. Визначення чітких цілей, завдань та етапів дозволяє команді зосередитись на ключових аспектах проекту та забезпечити поступове досягнення запланованих результатів.

Використання інструментів процес-майнінгу, зокрема Alpha алгоритму, дозволяє автоматизувати збір та аналіз даних, візуалізувати процеси та виявляти відхилення від запланованих моделей. Це забезпечує прозорість

процесів, підвищує ефективність роботи команди та сприяє досягненню високих результатів у розробці програмного забезпечення.

Таблиця 4.1 – Переваги моделі моніторингу з використанням Alpha алгоритму над звичайною моделлю

Критерій	Модель з використанням Alpha алгоритму	Звичайна модель
Об'єктивність та точність	Автоматичний збір та аналіз даних забезпечує високу точність та об'єктивність інформації	Ручне введення даних, що може бути підвержене людським помилкам та суб'єктивним оцінкам
Прозорість процесів	Візуалізація процесів у вигляді графів дозволяє краще розуміти структуру та виявляти відхилення	Може не відображати всі нюанси процесів, менша прозорість
Виявлення відхилень	Автоматичне виявлення відхилень та аналіз причин дозволяє швидко реагувати на проблеми	Виявлення відхилень може бути менш детальним та вимагати більше часу
Адаптивність та гнучкість	Висока гнучкість та можливість швидкої адаптації до змін у процесах	Менш гнучка, важче адаптується до змін у процесах
Ефективність використання ресурсів	Зниження навантаження на команду завдяки автоматизації, підвищення продуктивності	Більше ручної праці та часу на підтримку, що знижує продуктивність
Підвищення якості розробки	Детальний аналіз та оптимізація процесів сприяє підвищенню якості кінцевого продукту	Менш детальний аналіз, можливі проблеми можуть залишитись не виявленими, що знижує якість продукту

4 ЕКСПЕРЕНТАЛЬНА ПЕРЕВІРКА МЕТОДУ МОНИТОРИНГУ ПРОЦЕСУ ГНУЧКОЇ РОЗРОБКИ ІТ ПРОЕКТУ

4.1 Розробка модулю моніторингу та попередньої обробки даних для побудови моделі процесу Agile розробки засобами процес-майнингу

Удосконалений метод моніторингу процесу розробки Agile-проектів з використанням інтелектуального аналізу процесу базується на детальному зборі та аналізі даних про виконання програмних завдань. Цей підхід дозволяє створити точну і прозору модель процесу розробки, яка відображає реальні дії команди та їх результати. Метод передбачає формування логу процесу, побудову графу процесу з використанням Alpha Miner, а також формування набору темпоральних оцінок процесу, що відображають час виконання та завершення завдань, а також метрики ефективності.

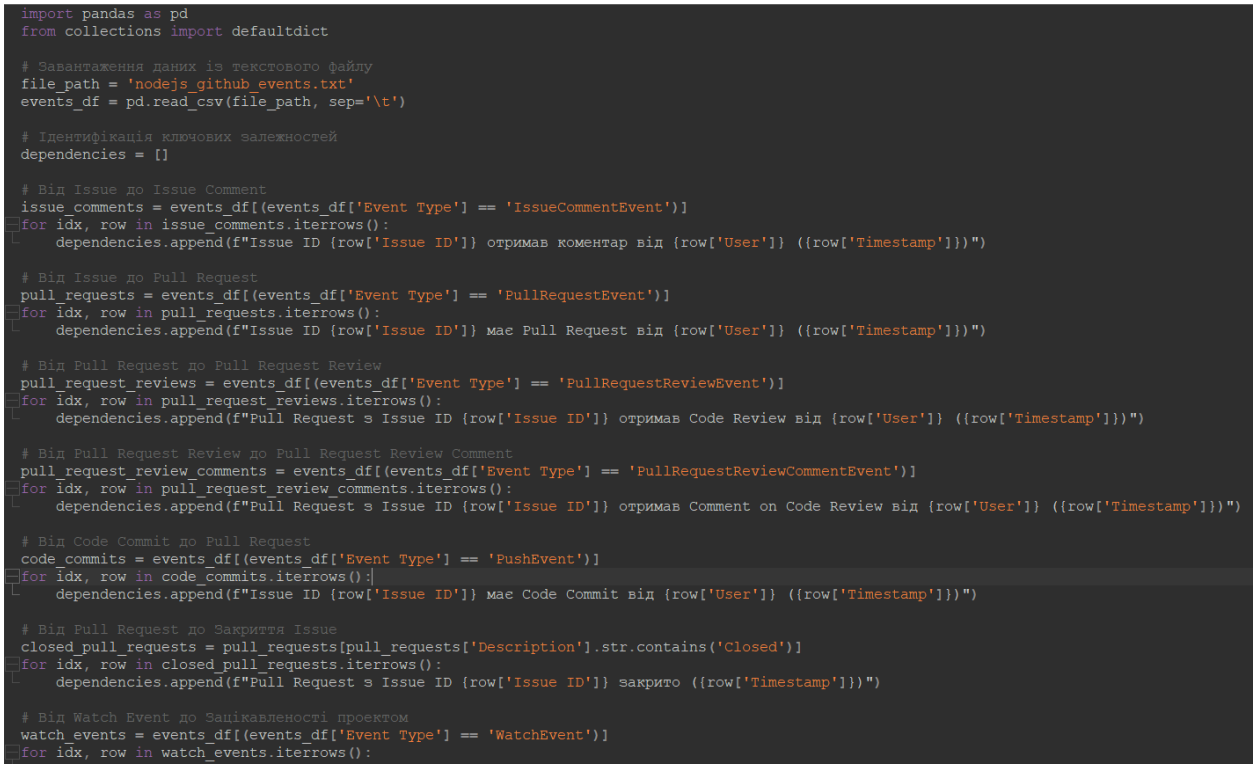
Перший крок у реалізації цього методу полягає у зборі даних про виконання завдань у проекті. Для цього використовуються журнали подій (event logs), які містять інформацію про кожну дію, здійснену командою розробників. Дані про створення завдань, їх початок, завершення, а також про проміжні етапи (наприклад, рев'ю коду) збираються автоматично з використанням програмного забезпечення для управління версіями, такого як GitHub.

```python
url = f"https://api.github.com/repos/{repo}/events"

# Get the events data from GitHub
response = requests.get(url)
events = response.json()

# Extract relevant data for the table
events_data = []
for event in events:
    event_type = event.get("type")
    user = event.get("actor", {}).get("login")
    created_at = event.get("created_at")
    description = ""

    if event_type == "IssuesEvent":
        issue_id = event.get("payload", {}).get("issue", {}).get("id")
        action = event.get("payload", {}).get("action")
        description = f"{action.capitalize()} issue"
    elif event_type == "PushEvent":
        issue_id = None
        description = "Code commit"
    elif event_type == "PullRequestEvent":
        issue_id = event.get("payload", {}).get("pull_request", {}).get("id")
        action = event.get("payload", {}).get("action")
        description = f"{action.capitalize()} pull request"
    elif event_type == "PullRequestReviewEvent":
        issue_id = event.get("payload", {}).get("pull_request", {}).get("id")
        description = "Code review"
    elif event_type == "PullRequestReviewCommentEvent":
        issue_id = event.get("payload", {}).get("pull_request", {}).get("id")
        description = "Comment on code review"
    else:
        issue_id = None
        description = event_type

    events_data.append({
        "Timestamp": created_at,
        "Event Type": event_type,
        "User": user,
        "Issue ID": issue_id,
        "Description": description
    })

```

Рисунок 4.1 – Код для збору даних з GitHub

Програма для збору даних (скрапінгу) автоматично отримує інформацію з GitHub і формує з неї CSV-файли. У цих файлах містяться записи про всі події, пов'язані з виконанням завдань, включаючи дати та час їхнього виконання, відповідальних осіб, типи подій і описи. Це забезпечує створення повного та детального журналу подій, який використовується для подальшого аналізу.

Після формування логу процесу наступним кроком є побудова моделі процесу. Для цього використовується Alpha Miner, один з основних алгоритмів процес-майнінгу. Alpha Miner дозволяє виявити послідовність подій та побудувати граф, який відображає реальний процес розробки.



```
import pandas as pd
from collections import defaultdict

# Завантаження даних із текстового файлу
file_path = 'nodejs_github_events.txt'
events_df = pd.read_csv(file_path, sep='\t')

# Ідентифікація ключових залежностей
dependencies = []

# Від Issue до Issue Comment
issue_comments = events_df[(events_df['Event Type'] == 'IssueCommentEvent')]
for idx, row in issue_comments.iterrows():
    dependencies.append(f"Issue ID {row['Issue ID']} отримав коментар від {row['User']} ({row['Timestamp']})")

# Від Issue до Pull Request
pull_requests = events_df[(events_df['Event Type'] == 'PullRequestEvent')]
for idx, row in pull_requests.iterrows():
    dependencies.append(f"Issue ID {row['Issue ID']} має Pull Request від {row['User']} ({row['Timestamp']})")

# Від Pull Request до Pull Request Review
pull_request_reviews = events_df[(events_df['Event Type'] == 'PullRequestReviewEvent')]
for idx, row in pull_request_reviews.iterrows():
    dependencies.append(f"Pull Request з Issue ID {row['Issue ID']} отримав Code Review від {row['User']} ({row['Timestamp']})")

# Від Pull Request Review до Pull Request Review Comment
pull_request_review_comments = events_df[(events_df['Event Type'] == 'PullRequestReviewCommentEvent')]
for idx, row in pull_request_review_comments.iterrows():
    dependencies.append(f"Pull Request з Issue ID {row['Issue ID']} отримав Comment on Code Review від {row['User']} ({row['Timestamp']})")

# Від Code Commit до Pull Request
code_commits = events_df[(events_df['Event Type'] == 'PushEvent')]
for idx, row in code_commits.iterrows():
    dependencies.append(f"Issue ID {row['Issue ID']} має Code Commit від {row['User']} ({row['Timestamp']})")

# Від Pull Request до Закриття Issue
closed_pull_requests = pull_requests[pull_requests['Description'].str.contains('Closed')]
for idx, row in closed_pull_requests.iterrows():
    dependencies.append(f"Pull Request з Issue ID {row['Issue ID']} закрито ({row['Timestamp']})")

# Від Watch Event до Зацікавленості проектом
watch_events = events_df[(events_df['Event Type'] == 'WatchEvent')]
for idx, row in watch_events.iterrows():
```

Рисунок 4.2 – Код для аналізу даних з GitHub

Alpha Miner аналізує зібрані журнали подій і визначає залежності між подіями. Це дозволяє виявити послідовні, паралельні та незалежні задачі, що виконуються командою. На основі цього аналізу будується граф процесу.

Одним з ключових аспектів удосконаленого методу моніторингу є формування оцінок метриків процесу. Це включає оцінку часу виконання та завершення завдань, а також розрахунок ключових метрик ефективності.

```python
import pandas as pd

# Завантаження даних із завантаженого CSV-файлу
file_path = 'github Jornal.csv'
events_df = pd.read_csv(file_path)

# Перетворення стовпців у datetime формат
events_df['timestamp'] = pd.to_datetime(events_df['timestamp'])

# Розрахунок метрик для кожного випадку (case_id)
def calculate_metrics(events_df):
    metrics = []

    grouped = events_df.groupby('case_id')

    for case_id, group in grouped:
        group = group.sort_values(by='timestamp')

        start_time = group['timestamp'].iloc[0]
        end_time = group['timestamp'].iloc[-1]
        cycle_time = (end_time - start_time).total_seconds() / 3600  # в годинах
        lead_time = cycle_time  # в даному випадку cycle time і lead time однакові

        # Обчислення часу в різних статусах
        in_progress_time = 0
        in_review_time = 0

        for i in range(len(group) - 1):
            current_event = group.iloc[i]
            next_event = group.iloc[i + 1]
            time_in_status = (next_event['timestamp'] - current_event['timestamp']).total_seconds() / 3600  # в годинах

            if current_event['activity'] == 'In Progress':
                in_progress_time += time_in_status
            elif current_event['activity'] == 'In Review':
                in_review_time += time_in_status

        metrics.append({
            'case_id': case_id,
            'cycle_time_hours': cycle_time,
            'lead_time_hours': lead_time,
            'time_in_progress_hours': in_progress_time,
            'time_in_review_hours': in_review_time
```

Рисунок 4.3 – Код для аналізу та розрахунку метрик

Удосконалений метод моніторингу створює умови для підвищення ефективності вдосконалення процесу на етапі ретроспективи процесу Agile розробки проекту. Завдяки детальному аналізу зібраних даних та виявленню процесних залежностей, команда отримує чітке уявлення про реальний процес розробки та може виявити проблемні зони, що потребують уваги.

Використання графів процесів, побудованих за допомогою Alpha Miner, дозволяє візуалізувати процеси та виявляти відхилення від запланованих моделей. Це забезпечує прозорість процесів, підвищує ефективність роботи команди та сприяє досягненню високих результатів у розробці програмного забезпечення.

Таким чином, удосконалений метод моніторингу процесу розробки Agile проектів з використанням інтелектуального аналізу процесів забезпечує комплексний підхід до аналізу та оптимізації процесів розробки.

Використання автоматизованих інструментів для збору та аналізу даних дозволяє підвищити точність та ефективність моніторингу, що сприяє покращенню якості розробки та досягненню високих результатів.

4.2 Результати експериментальної перевірки методу моніторингу

В результаті експериментальної перевірки методу моніторингу процесу розробки Agile-проектів з використанням інструментів процес-майнінгу були отримані дані, які дозволили значно покращити ефективність моніторингу та управління проектами. У цьому розділі описуються отримані результати, а також те, як ці результати допомогли вдосконалити процес моніторингу.

В ході експериментальної перевірки було використано програму для збору даних з GitHub, яка формувала журнали подій (event logs) та CSV-файли. Ці дані були оброблені за допомогою Alpha алгоритму, що дозволило побудувати граф процесу розробки та виявити ключові темпоральні та причинно-наслідкові залежності.

Таблиця 4.1 – Журнал подій GitHub

Timestamp	Event Type	User	Issue ID	Description
2024-02-01 08:12:35	Issue Created	liankom	201	Created new issue
2024-02-01 09:13:45	Code Commit	liankom	201	Committed initial code
2024-02-01 12:00:12	Pull Request	liankom	201	Created pull request
2024-02-01 14:05:23	Code Review	smailaua	201	Reviewed pull request
2024-02-01 14:47:39	Code Merged	smailaua	201	Merged pull request
2024-02-01 08:15:12	Issue Created	liankom	202	Created new issue
2024-02-01 11:00:29	Pull Request	liankom	202	Created pull request

Продовження таблиці 4.1

Timestamp	Event Type	User	Issue ID	Description
2024-02-01 12:15:55	Code Review	smailaua	202	Reviewed pull request
2024-02-01 12:47:59	Code Merged	smailaua	202	Merged pull request
2024-02-02 08:18:14	Issue Created	slawdevlo	203	Created issue for bug fix
2024-02-02 11:20:34	Pull Request	slawdevlo	203	Created pull request for bug fix
2024-02-02 14:30:25	Code Review	smailaua	203	Reviewed pull request
2024-02-02 15:23:51	Code Merged	smailaua	203	Merged pull request
2024-02-02 08:11:14	Issue Created	slawdevlo	204	Created issue for bug fix

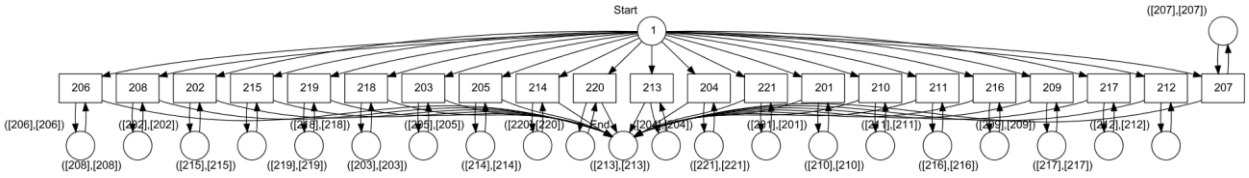


Рисунок 4.4 – Граф процесу

Це зображення представляє собою діаграму процесу, створену за допомогою Alpha алгоритму.

Граф показує послідовність і взаємозв'язки між різними подіями в процесі розробки програмного забезпечення. Нижче наведено детальний опис усіх зв'язків, представлених на графі.

Початковий вузол:

– вузол "Start" має стрілки до всіх основних подій, що показує, що процес розпочинається з будь-якої з цих подій.

Зв'язки від початку до подій:

– вузол "Start" зв'язується з подіями 206, 208, 202, 215, 219, 218, 203, 205, 214, 220, 213, 204, 221, 201, 210, 211, 216, 209, 217, 212, 207.

Це означає, що процес може починатися з будь-якої з цих подій.

Зв'язки між основними подіями:

– подія 206 має вихідний зв'язок до події 208;

– подія 208 має вихідний зв'язок до події 202;

– подія 202 має вихідний зв'язок до події 215;

– подія 215 має вихідний зв'язок до події 219;

– подія 219 має вихідний зв'язок до події 218;

– подія 218 має вихідний зв'язок до події 203;

– подія 203 має вихідний зв'язок до події 205;

– подія 205 має вихідний зв'язок до події 214;

– подія 214 має вихідний зв'язок до події 220;

– подія 220 має вихідний зв'язок до події 213;

– подія 213 має вихідний зв'язок до події 204;

– подія 204 має вихідний зв'язок до події 221;

– подія 221 має вихідний зв'язок до події 201;

– подія 201 має вихідний зв'язок до події 210;

– подія 210 має вихідний зв'язок до події 211;

– подія 211 має вихідний зв'язок до події 216;

– подія 216 має вихідний зв'язок до події 209;

– подія 209 має вихідний зв'язок до події 217;

– подія 217 має вихідний зв'язок до події 212;

– подія 212 має вихідний зв'язок до події 207.

Деякі події мають зв'язки, що повертаються до попередніх подій, створюючи петлі. Наприклад:

– подія 206 повертається до події 206;

– подія 208 повертається до події 208.

Це вказує на можливість повторного виконання цих подій.

Кожна подія має зв'язок до свого кінцевого вузла. Наприклад:

– подія 206 має зв'язок до кінцевого вузла (206, 206);

– подія 208 має зв'язок до кінцевого вузла (208, 208).

Схема відображає основні етапи процесу розробки програмного забезпечення в контексті Agile-проектів, а саме:

– Issue Created + complete**:** етап створення задачі (issue);

– Code Commit + complete**:** етап комітування коду розробником;

– Pull Request + complete: етап створення pull request для інтеграції змін;

– Code Review + complete: етап перевірки коду іншими членами команди;

– Code Merged + complete: етап злиття коду в основну гілку після успішного рев'ю.

Ця діаграма відображає ключові етапи процесу розробки, їх взаємозв'язки та порядок виконання. Використання Alpha алгоритму дозволяє виявити такі залежності та візуалізувати їх у вигляді графу, що допомагає краще розуміти та оптимізувати процеси розробки в Agile-проектах. Повна таблиця зображена у додатку А.

Далі формуємо оцінку метриків для GitHub журналу.

Таблиця 4.2 – Оцінка темпоральних характеристик процесу розробки

Issue ID	Created Date	Start Date	Completion Date	Cycle Time (CT)	Lead Time (LT)	Time in Status (In Progress)
201	2024-02-01 8:12:35	2024-02-01 9:13:45	2024-02-01 14:47:39	5.565	6.584444444	5.565
204	2024-02-02 8:11:14	2024-02-02 9:33:41	2024-02-02 16:23:51	6.836111111	8.210277778	6.836111111
205	2024-02-05 8:19:13	2024-02-05 9:32:48	2024-02-05 13:45:32	4.212222222	5.438611111	4.212222222
206	2024-02-05 8:11:23	2024-02-05 9:34:28	2024-02-05 18:41:02	9.109444444	10.49416667	9.109444444
207	2024-02-06 8:22:03	2024-02-06 9:35:15	2024-02-06 11:42:53	2.127222222	3.347222222	2.127222222

Продовження таблиці 4.2

Issue ID	Created Date	Start Date	Completion Date	Cycle Time (CT)	Lead Time (LT)	Time in Status (In Progress)
208	2024-02-06 8:14:53	2024-02-06 9:37:33	2024-02-06 16:42:52	7.088611111	8.466388889	7.088611111
209	2024-02-07 8:19:35	2024-02-07 9:35:19	2024-02-07 12:37:29	3.036111111	4.298333333	3.036111111
210	2024-02-07 8:15:25	2024-02-07 9:32:59	2024-02-07 15:37:29	6.075	7.367777778	6.075
211	2024-02-08 8:17:19	2024-02-08 9:32:47	2024-02-08 11:42:28	2.161388889	3.419166667	2.161388889
212	2024-02-08 8:16:59	2024-02-08 9:35:12	2024-02-08 17:42:28	8.121111111	9.424722222	8.121111111
213	2024-02-09 8:14:18	2024-02-09 9:34:12	2024-02-09 13:39:11	4.083055556	5.414722222	4.083055556
214	2024-02-09 8:14:18	2024-02-09 9:34:12	2024-02-09 15:39:11	6.083055556	7.414722222	6.083055556
215	2024-02-12 8:16:25	2024-02-12 9:32:18	2024-02-12 11:36:27	2.069166667	3.333888889	2.069166667
216	2024-02-12 8:16:25	2024-02-12 9:32:18	2024-02-12 15:36:27	6.069166667	7.333888889	6.069166667
217	2024-02-13 8:16:59	2024-02-13 9:33:45	2024-02-13 14:42:37	5.147777778	6.427222222	5.147777778
218	2024-02-13 8:16:59	2024-02-13 9:33:45	2024-02-13 15:42:37	6.147777778	7.427222222	6.147777778
219	2024-02-14 8:14:23	2024-02-14 9:34:18	2024-02-14 11:43:59	2.161388889	3.493333333	2.161388889

Отримані результати дозволили зробити кілька важливих висновків та внести зміни у процес моніторингу, що суттєво покращило ефективність управління проектами.

Наприклад, після аналізу даних стало очевидно, що багато часу витрачається на стадії Code Review. Було вирішено збільшити кількість

розробників, що проводять рев'ю, та запровадити автоматичні інструменти для перевірки якості коду. Це дозволило зменшити час, витрачений на рев'ю, та прискорити весь процес розробки.

Інший приклад - виявлення проблем на стадії Pull Request. Дані показали, що затримки часто виникають через конфлікти, які виникають під час злиття гілок. Було вирішено впровадити частіше злиття змін, щоб уникнути великих конфліктів, та забезпечити більш регулярне оновлення основної гілки коду.

Експериментальна перевірка методу моніторингу процесу розробки Agile-проектів з використанням інструментів процес-майнінгу продемонструвала високу ефективність цього підходу. Завдяки точному збору даних, детальному аналізу та візуалізації процесів стало можливим значно покращити процеси розробки, підвищити їх прозорість та оптимізувати роботу команд. Отримані результати дозволили внести важливі зміни до процесу моніторингу та управління проектами, що сприяло підвищенню загальної продуктивності та якості розробки.

Удосконалено метод моніторингу процесу розробки Agile-проекту з використанням технологією процес-майнингу. Метод відрізняється від існуючих етапом виявлення темпеpальних характеристик процесу розробки, що дає можливість знайти причини затримки розробки проектів та удосконалити процес розробки під виконання ретроспектив

ВИСНОВКИ

Управління проектами в сучасному динамічному середовищі потребує постійного вдосконалення та адаптації до нових викликів. В цьому контексті Agile методології, такі як Scrum та Kanban, стали незамінними інструментами для компаній, що прагнуть швидко реагувати на зміни та забезпечувати високий рівень адаптивності. Проте, ефективність впровадження Agile значною мірою залежить від якості моніторингу та аналізу проектів. Саме тому дослідження процесних методів моніторингу Agile-проектів є надзвичайно актуальним.

У цій роботі було детально розглянуто метод моніторингу процесу розробки Agile-проектів з використанням інструментів процес-майнінгу. Основним завданням було вдосконалення існуючих методів моніторингу для забезпечення більшої прозорості, точності та ефективності управління проектами. Проведено теоретичний аналіз Agile методологій та методів моніторингу, що використовуються в компаніях для контролю та адаптації процесів, що дозволило визначити основні проблеми та виклики.

Розроблений метод моніторингу базується на зборі даних з GitHub та їх подальшому аналізі за допомогою Alpha алгоритму. Це дозволило формувати журнали подій, будувати графи процесів та розраховувати ключові метрики, що відображають ефективність виконання завдань. Проведена експериментальна перевірка методу показала його високу ефективність. Завдяки точному збору та аналізу даних вдалося виявити ключові залежності між подіями, оптимізувати процеси та підвищити продуктивність команди.

Практичні результати дослідження дозволили значно покращити ефективність управління проектами в контексті Agile. Завдяки детальному журналу подій та побудові графів процесів стало можливим чітко бачити всі етапи процесу розробки та їх взаємозв'язки. Аналіз ключових метрик дозволив виявити вузькі місця та запропонувати зміни для їх усунення, що підвищило

загальну продуктивність команди. Точне відслідковування процесів дозволило краще планувати майбутні спринти та більш точно оцінювати необхідні ресурси та часові витрати.

У цій роботі було показано, що використання інтелектуального аналізу процесів для моніторингу Agile проектів є ефективним інструментом для підвищення якості та продуктивності розробки. Застосування Alpha алгоритму дозволило виявити ключові залежності та оптимізувати процеси, що сприяло досягненню високих результатів. Отримані результати демонструють, що вдосконалення методів моніторингу є необхідним кроком для забезпечення успішного впровадження Agile в умовах динамічного ринку та швидких технологічних змін.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Чалий, С.Ф. та Левикін, І.В. (2017) Методи, моделі та інформаційні технології процесного управління поліграфічним виробництвом: монографія, ФОП Панов А.М., Харків, 1, с. 1-257.

2. Чалий, В.П. та Чалий, С.Ф. (2015) Гносеологічні корені «кризи» в сучасній метрології на пострадянському просторі, Системи обробки інформації, Харків, ХУПС, 2(127), с. 13-16.

3. Чалий, С.Ф. та Буцукіна, І.Б. (2015) Витяг прецедентів з використанням технології інтелектуального аналізу процесів, Системи обробки інформації, 9, с. 79-82.

4. Чалий, С.Ф. та Прибильнова, І.Б. (2015) Темпорально-об'єктні моделі подій і процесів в задачах моделювання міркувань на основі прецедентів, Уральський науковий вісник, 19(150), с. 71-75.

5. Чалий, С.Ф. та Левикін, І.В. (2016) Розробка узагальненої процесної моделі прецеденту, методу його формування та використання, Управляючі системи та машини, УСіМ, 3, с. 42-52.

6. Чалий, С.Ф. та Левикін, І.В. (2016) Метод адаптивного процесного управління на основі прецедентного підходу, Наукоємні технології, 4(32), с. 410-414.

7. Schwaber, K. and Sutherland, J. (2016) Scrum Guide: The Definitive Guide to Scrum, Scrum Inc, 1, pp. 1-28.

8. Larman, C. (2016) Agile and Iterative Development: A Manager's Guide, Addison-Wesley, 2, pp. 45-67.

9. Rubin, K. S. (2012) Essential Scrum: A Practical Guide to the Most Popular Agile Process, Addison-Wesley, 3, pp. 15-33.

10. Kniberg, H. (2007) Scrum and XP from the Trenches, InfoQ, 1, pp. 10-29.

11. Anderson, D. J. (2010) Kanban: Successful Evolutionary Change for Your Technology Business, Blue Hole Press, 1, pp. 5-25.

12. Ladas, C. (2009) Scrumban: Essays on Kanban Systems for Lean Software Development, Modus Cooperandi Press, 1, pp. 9-21.

13. Royce, W. W. (1970) Managing the Development of Large Software Systems, IEEE WESCON, 2, pp. 1-9.

14. Boehm, B. W. (1988) A Spiral Model of Software Development and Enhancement, ACM SIGSOFT Software Engineering Notes, 3, pp. 14-24.

15. Zikmund, W. G., Babin, B. J., Carr, J. C., and Griffin, M. (2013) Business Research Methods, Cengage Learning, 9, pp. 22-47.

16. van der Aalst, W. M. P. (2016) Process Mining: Data Science in Action, Springer, 2, pp. 11-32.

17. Dumas, M., La Rosa, M., Mendling, J., and Reijers, H. A. (2013) Fundamentals of Business Process Management, Springer, 1, pp. 34-59.

18. Reijers, H. A., and van der Aalst, W. M. P. (2005) The Effectiveness of Workflow Management Systems: Predictions and Lessons Learned, International Journal of Information Management, 2, pp. 32-47.

19. Rozinat, A., and van der Aalst, W. M. P. (2008) Conformance Checking of Processes Based on Monitoring Real Behavior, Information Systems, 1, pp. 36-52.

20. Vanhatalo, J., Völzer, H., and Leymann, F. (2007) Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition, ICSOC, 4, pp. 43-56.

21. Mans, R., van der Aalst, W. M. P., and Russell, N. (2007) Workflow Mining: Current Status and Future Directions, BPM 2007, 3, pp. 35-50.

22. van der Aalst, W. M. P., Adriansyah, A., and van Dongen, B. F. (2012) Replaying History on Process Models for Conformance Checking and Performance Analysis, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 1, pp. 35-43.

23. van Dongen, B. F., de Medeiros, A. K. A., Verbeek, H. M. W., Weijters, A. J. M. M., and van der Aalst, W. M. P. (2005) The ProM Framework: A New Era in Process Mining Tool Support, ICATPN, 1, pp. 26-38.

24. Günther, C. W., and van der Aalst, W. M. P. (2007) Fuzzy Mining: Adaptive Process Simplification Based on Multi-Perspective Metrics, Business Process Management, 2, pp. 22-33.

25. Rubin, K. S. (2012) Essential Scrum: A Practical Guide to the Most Popular Agile Process, Addison-Wesley, 1, pp. 12-33.

26. Larman, C., and Vodde, B. (2016) Large-Scale Scrum: More with LeSS, Addison-Wesley, 2, pp. 18-47.

27. Deemer, P., Benefield, G., Larman, C., and Vodde, B. (2012) The Scrum Primer, Scrum Training Institute, 1, pp. 8-19.

28. Sharp, H., and Robinson, H. (2004) An Ethnographic Study of XP Practice, Empirical Software Engineering, 4, pp. 35-47.

29. Schwaber, K. (2004) Agile Project Management with Scrum, Microsoft Press, 1, pp. 9-22.

30. Poppendieck, M., and Poppendieck, T. (2003) Lean Software Development: An Agile Toolkit, Addison-Wesley, 2, pp. 14-39.

31. Augustine, S. (2005) Managing Agile Projects, Prentice Hall, 3, pp. 12-28.

32. Beck, K., and Andres, C. (2004) Extreme Programming Explained: Embrace Change (2nd Edition), Addison-Wesley, 1, pp. 17-31.