Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра Інформаційних управляючих систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження процесних методів гнучкої розробки програмного забезпечення інформаційних систем
(тема)

Виконала:
студентка _2_ курсу, групи _ІУСТзм-22-1__
Горбовцова Інна Вікторівна
(прізвище, ім'я, по батькові)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма _Інформаційні управляючі системи та технології
(повна назва освітньої програми)
Керівник Сергій ЧАЛИЙ
(Власне ім'я ПРІЗВИЩЕ)

Допускається до захисту

Зав. кафедри _____ Костянтин ПЕТРОВ
(підпис) (Власне ім'я ПРІЗВИЩЕ)

2024 р.

Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання

Кафедра Інформаційних управляючих систем

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

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

Освітня програма Інформаційні управляючі системи та технології
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«_04_» _грудня_ 20 23_ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Горбовцовій Інні Вікторівні
(прізвище, ім'я, по батькові)

1. Тема роботи _Дослідження процесних методів гнучкої розробки програмного забезпечення інформаційних систем_____
затверджена наказом університету від _01_ __грудня__2023_ р. № _259Стз_
2. Термін подання студентом роботи до екзаменаційної комісії __19__ _січня___ 2024_ р.
3. Вихідні дані до роботи _Модель логу розробки; проаналізований лог розробки; визначені вузькі місця логу розробки; експериментальна перевірка._____
4. Перелік питань, що потрібно опрацювати в роботі: Аналіз процесу розробки програмного забезпечення інформаційних систем, аналіз Agile методів; дослідження процесних методів управління, темпоральний опис процесу розробки програмного забезпечення інформаційних систем, удосконалення методу оцінки процесу розробки програмного забезпечення інформаційних систем із використанням темпоральних залежностей, технологія оцінки процесу розробки програмного забезпечення з використанням темпоральних залежностей, реалізація та експериментальна перевірка методу оцінки процесу розробки програмного забезпечення інформаційних систем.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання	04.12.2023	виконано
2	Аналіз процесу розробки програмного забезпечення інформаційних систем	04.12.2023 – 08.12.2023	виконано
3	Аналіз гнучких методів та підходів й аналіз процесних методів	08.12.2023 – 12.12.2023	виконано
4	Дослідження темпоральних правил	12.12.2023 – 16.12.2023	виконано
5	Експериментальна перевірка	16.12.2023 – 25.12.2023	виконано
6	Оформлення пояснювальної записки та графічного матеріалу	04.01.2024 – 16.01.2024	виконано
7	Захист кваліфікаційної роботи	22.01.2024	виконано

Дата видачі завдання _04_ ___грудня___2023 р.

Студент _____ _____Інна ГОРБОВЦОВА_____
(підпис) (власне ім’я, прізвище)

Керівник роботи _____ __проф. каф. ІУС Сергій ЧАЛИЙ__
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи містить: 99 с., 4 розділи, 27 рис., 8 табл., 1 додаток, 24 джерела.

ГНУЧКИЙ МЕТОД, ІНФОРМАЦІЙНІ СИСТЕМИ, ІТЕРАЦІЯ, ПРОЦЕСНІ МЕТОДИ, ТЕМПОРАЛЬНІ ПРАВИЛА, AGILE, SCRUM.

Об’єкт дослідження – процес аналізу логів подій.

Предмет дослідження – процесні методи гнучкої розробки програмного забезпечення інформаційних систем.

Метою даної роботи є дослідження логів подій на етапі ретроспективи на основі процесних методів та темпоральних правил.

В ході роботи було проведене дослідження процесних методів гнучкої розробки програмного забезпечення інформаційних систем. Проаналізовано методи гнучкої розробки інформаційних систем. Проаналізовано процесні методи. За допомогою темпоральних правил удосконалено метод аналізу на етапі ретроспективи. Проведена експериментальна перевірка.

ABSTRACT

Explanatory note to the qualification work: 99 p., 4 chapters, 27 figures, 8 tables, 1 appendix, 24 sources.

FLEXIBLE METHOD, INFORMATION SYSTEMS, ITERATION, PROCESS METHODS, TEMPORAL RULES, AGILE, SCRUM.

The object of research is the process of analyzing event logs.

The subject of research is process methods of flexible development of information systems software.

The purpose of this work is to study event logs at the retrospective stage based on process methods and temporal rules.

The research of process methods of flexible software development of information systems is carried out in the work. The methods of flexible development of information systems are analyzed. Process methods are analyzed. With the help of temporal rules, the method of analysis at the retrospective stage has been improved. An experimental test was carried out.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ІС – Інформаційні системи

ПЗ – Програмне забезпечення

ХР – Екстримальне програмування

XES – Extensible Event Stream

CSV – Comma-Separated Values

ВСТУП

Із кожним днем людство все більше погоджується з тим фактом, що ІТ сьогодні - одна з найбільш розвинених сфер . Тому не дивно, що більшість компаній залюбки застосовують у своїй роботі інформаційні технології для отримання конкурентних переваг, а також активно впроваджують ІТ- проєкти, які отримали свою популярність завдяки розвитку інформаційних технологій, появі різних видів програмного забезпечення, автоматизації діяльності різних організацій.

Величезний попит на ІТ-спеціалістів спричинив збільшення обсягів роботи в цій сфері. Однак, щоб збільшити обсяг роботи завдань необхідно краще організувати робочий процес створення програмного продукту у сфері інформаційних технологій. Щоб досягти успіху, потрібно мати систему розподілу завдань, вести звітність, це буде допомагати відстежувати продуктивність команди. Існує дуже багато практик, корисних компаніям, що спеціалізуються на вирішенні проблеми використання методологій і моделей розподілу робіт, планування, створення нових рішень на етапі ретроспективи.

Важко уявити, що колись робота з розробки проєктів не мала таких систем і організацій, що є на сьогоднішній день. Якщо замислитися, скільки усього було створено за усю історію цивілізованого світу, то, мабуть, що набереться декілька тисячоліть досвіду не одного покоління по реалізації проєктів, де можна отримати відповідний урок. Якщо провести паралель від будівників єгипетських пірамід або проєктувальників римських акведуків протягом століть до сучасного ІТ-спеціаліста, то можна помітити, що у всі часи керівники проєктів виконувади схожі ролі, вирішували характерні проблеми для своїх ролей. Навіть у сьогодення, коли більшість виконавців

хочуть удосконалювати методи управління розробки програм і веб-додатків, вони дуже рідко звертають свою увагу на уроки, які були пройдені ще тисячоліття тому.

Наразі розробка інформаційних є тим процесом, котрий потребує у першу чергу вирішення пролем з показником часу, бюджету та під кінець самої функціональності розроблюваного програмного продукту. Виникають випадки, коли проєкти реалізуються запізно. Тож головним завданням залишається те, що потрібно знайти баланс між цими трьома частинами (бюджет, час і розробка), щоб задовольнити вимоги та очікування клієнта, забезпечивши успішне завершення проєкту.

Одним із найефективніших інструментів для аналізу логів подій, які зараз активно використовують компанії стали процесні методи. Саме вони стали тією областю досліджень, де необхідно проводити аналіз процесів із використанням логів.

1 АНАЛІЗ ПРОЦЕСУ ГНУЧКОЇ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Дослідження процесу розробки програмного забезпечення інформаційних систем

Однією із основних проблем у сфері інформаційних технологій є управління розробкою програмного забезпечення. Тому на сьогоднішній день для того, щоб ефективно створювати програмні продукти необхідно не лише займатися створенням коду, а й звернути увагу на 4 основні фази:

– ініціація;
– планування;
– виконання;
– завершення.

Під час ініціації (початкової фази) відбувається формування концепції проєкту, де визначено вихідні цілі, розглянуто питання можливості реалізації, обґрунтовано актуальність, альтернативи, виділено бюджет, необхідні ресурси, а також призначено відповідального за проєкт (керівника) та команду спеціалістів. Перехід до наступної фази відбувається лише після погодження керівництва та ухвалення рішення про настання початку виконання робіт.

Під час планування фіксується увага на таких етапах: постановка задач проєкту, визначення залежностей, визначення питання, яке стосується часу, визначення технічних , апаратних ресурсів, а ще стратегії розробки. Керівник проєкту з’ясовує питання з наявності необхідних ресурсів, узгоджує бюджет проєкту, а також відповідає за розподіл встановлених завдань.

Етап виконання починається після затвердження плану. Починається виконання плану проєкту, тестуються функції продуктивності згідно з

графіком проєкту. На даному етапі керівник проєкту здійснює контроль над роботою проєкту, а також відповідає за комунікації з замовником, командою. Керівнику проєкту необхідно порівнювати проєктний графік, який є визначений планом із фактичними показниками робіт, технічним завданням проєкту, аналізувати змінність впливу обсягу робіт, адаптувати й створювати корективи у розробці відповідних управлінських рішень, де звіти стану проєкту будуть висвітлювати питання бюджету очікуваного програмного продукту, проєктного графіку, а також якості розроблюваного проєкту.

Під час етапу завершення замовнику надаються кінцеві результати, документація по проєкту, випуск ресурсів проєкту, передача закриття проєкту для зацікавлених сторін.

Рисунок 1.1 – Фази життєвого циклу

Варто зазначити, що під час реалізації проєкт проходить різні фази, котрі у своїй сукупності й називають життєвим циклом. Також сюди можна вписати

аналіз предметної області, визначення вимог, розробку рішень, проєктування, документацію, тестування, контроль, впровадження, експлуатацію, обслуговування та управління змінами. Ці фази можуть змінюватися відповідно потребам замовника та специфіки розроблюваного програмного продукту.

У 1968 році відбулася конференція НАТО, на якій було розглянуто питання максимальної продуктивності комп'ютерів Л. Бауером. Після цього було створено стандарти, регламенти, методи і «кращі практики», які в сукупності таких практик, застосованих на різних стадіях життєвого циклу ПЗ, отримали назву «Методологія розробки програмного забезпечення». Так, методології розробки ПЗ стали однією із найрозвинутіших областей знань, адже вони несуть у собі практичну складову.

Щоб розробити якісне програмне забезпечення, належить зрозуміти основні принципи життєвого циклу ПЗ, вимоги замовника до кінцевого продукту. Необхідно враховувати і фінансові можливості. Існують різні моделі життєвого циклу проєкту (каскадна, спіральна, ітеративна тощо). Вибір конкректної моделі в цілому залежить від характеру та цілей щодо створюваного програмного продукту.

Каскадна модель або водоспадна модель (Waterfall model) є традиційним підходом до управління щодо розробки програмного забезпечення інформаційних систем. Waterfall model являє собою процес, де кожна фаза має свій набір завдань, а проєкт тільки по завершенню однієї фази переходить до наступної.

Важливо врахувати, що каскадна модель має як позитивні, так і негативні сторони. Нижче представлена таблиця переваг та недоліків каскадної моделі (Waterfall).

Таблиця 1.1 – Переваги і недоліки каскадної моделі

Переваги каскадної моделі	Недоліки каскадної моделі
Перевагою є простота структури процесу, що надає легшого розуміння і виконання завдань командою.	Недоліком є неможливість у гнучкості (даний метод не передбачає гнучкість під час розробки).
Також до переваг відноситься зручна звітність , адже вона дозволяє легко відстежувати ресурси/ризики/час/фінанси.	Важкість внесення змін, адже під час виникнення потреби змін у проєкті вони стають складними та накладними.
Перевагою є стабільність завдань (визначається на початку розробки і залишається незмінним протягом усього процесу створення й впровадження програмногопродукту).	Недоліком є інерційність в розробці(виникнення неможливості змін формату).
До переваг відносять оцінку вартості і строків.	Недоліком є те, що тестування проводиться тільки після завершення всіх етапів розробки. Це часто призводить до виявлення проблем у подальшому.

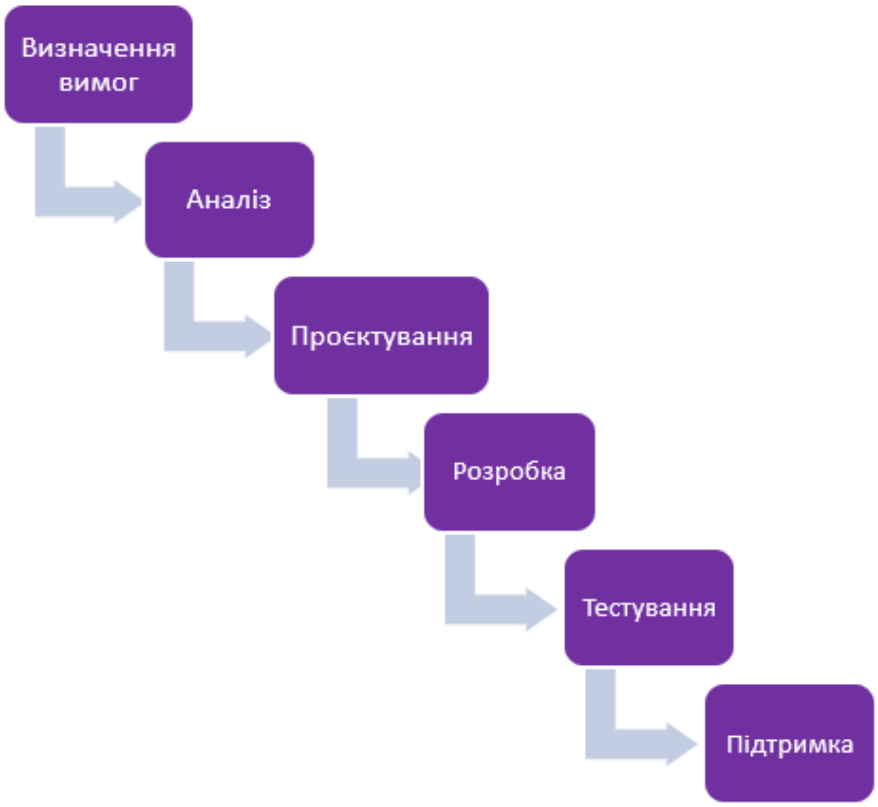

Рисунок 1.2 – Каскадна модель

Уолкер описав процес створення програмного продукту, що складається з шести ключових етапів. У 1985 році Міністерство внутрішньої безпеки США внесло цей процес до стандартів взаємодії з розробниками ПЗ:

– системні та програмні вимоги зафіксовані у спеціальному документі, відомому як PRD (Product Requirements Document);

– на етапі аналізу вимоги ретельно аналізуються, розробляються моделі, діаграми та бізнес-правила;

– на етапі проєктування розробляється внутрішня архітектура програмного продукту та способи реалізації вимог; це стосується не тільки зовнішнього вигляду та інтерфейсу продукту, але й внутрішньої структурної логіки;

– на етапі розробки пишеться програмний код та інтегрується ПЗ;

– на етапі тестування перевіряється кінцевий продукт, виявляються і документуються будь-які помилки в програмному коді або функціональності, будь-які знайдені помилки виправляються, якщо дозволяють час, бюджет;

– заключний етап підтримки передбачає адаптацію програмного продукту до різних операційних систем, окрім цього, регулярно оновлюється програма для виправлення помилок, виявлених користувачами, і додавання нових можливостей, також на цьому етапі надається технічна підтримка для задоволення потреб клієнтів.

Цікаво зазначити, що Toyota, відома своїми виробничими системами Lean та Kanban, прийняла каскадну модель розробки програмного забезпечення для виробничих потреб аж до кінця 2000-х років.

Методологія Waterfall тримається на низці основних принципів:

– послідовність означає, що робота проходить через серію послідовних етапів, де кожен наступний етап починається лише після іншого; це означає, що кожен етап має чітко визначений набір дій, такий підхід забезпечує структурований робочий процес;

– жорсткість кожного етапу, наприклад, визначення вимог, проєктування, розробка, тестування тощо має чітко визначені цілі та завдання (запобігає непорозумінням і забезпечує структурований підхід);

– деталі та вимоги мають бути чітко зафіксовані в документах заздалегідь (зазвичай з обмеженими змінами); чіткий підхід до специфікації може забезпечити стабільність і передбачуваність процесу розробки;

– великий обсяг робіт робить його придатним для великих проєктів, де вимоги можуть бути чітко визначені до початку (можна розділити на окремі етапи для кращого та ефективнішого управління);

– зміни вимог під час розробки, як правило, неприпустимі(кожен етап оснований на іншому, будь-яка зміна може призвести до переробки частин проєкту);

– етапи проєкту повинні бути чітко сплановані, а також належним чином відзвітовані та контрольовані(щоб допомогти контролювати прогрес і дотримання графіка).

Ітеративна модель життєвого циклу суттєво відрізняється від моделі каскаду. Цей підхід передбачає серію циклічних ітерацій, кожна фаза складається з чотирьох фаз: збір вимог, планування, розробка та тестування. Саме ця модель підкреслює важливість активної співпраці з клієнтом або користувачем протягом кожної ітерації. Замовник може побачити результати розробки на ранній стадії та внести зміни або уточнити вимоги.

Рисунок 1.3 – Ітеративна модель

Таблиця 1.2 – Переваги і недоліки ітеративної моделі

Переваги ітеративної моделі	Недоліки ітеративної моделі
До переваг відносять можливість значної взаємодії замовника з командою	Недоліком є те, що потрібно увесь час тримати усе під контролем в управлінні
Перевагою є швидке впровадження внесення змін (випрвалень)	До недоліків відносять вимогу у значній кількості наданих ресурсів
Перевага у сприянні в розробці більш адаптивних та відповідних продуктів	До недоліків відносять вірогідність у збільшенні витрат
Перевагою є можливість на ранніх етапах виявити проблему	При використанні даної моделі необхідно усе виконувати в строк
Перевагою є забезпечення можливості до змін пріоритетності та завдань в цілому	Недоліком є потреба тримати під контролем усю документацію

Важливою особливістю ітеративної моделі є те, що кожна ітерація додає нову функціональність до програмного продукту. Нова ітерація починається до завершення існуючої і тим самим розширює поточну версію програмного забезпечення. У цьому випадку напрямок проєктної діяльності можна змінювати на початку кожної ітерації, що може бути актуальним у разі швидкої зміни зовнішнього середовища. Функціональність усієї системи можна глобально оцінити на початку проєкту (на перших ітераціях), а незначні необхідні зміни можна внести на наступних. Окрім того, ітерації можуть слугувати інструментом зворотного зв’язку для вимірювання якості проєкту та продуктивності залученої команди. Ітеративний підхід до планування

життєвого циклу означає, що різні типи роботи над проєктом не пов’язані з конкретними фазами розробки; натомість вони виконуються за потреби.

Ця модель припускає, що розробку розуміють як послідовність ітерацій, кожна з яких сама по собі є невеликим проєктом у рамках загального завдання та призводить до вимірного приросту цінності продукту після завершення ітерації. На відміну від спіральної моделі, ітеративна методологія зміщує фокус із забезпечення повноти вимог до продукту на розробку процесу координації роботи в команді.

Потрібно зауважити, що модель була вперше впроваджена Вольтером Стюартом (спеціалістом із проблем якості) ще в 1930-х роках. Якщо опиратися на думку учасників проєкту, то ітеративний підхід був одним із ключових факторів успіху. Ітеративно-інкрементний підхід почав активно розвиватися в 1970-1980 роках. Одним із перших проєктів, де використовувалася ця модель, був гіперзвуковий літак X-15. Пізніше Iterative and Incremental development знайшла застосування у розробці різноманітних проєктів , послужила витоком для інших моделей та методологій. Варто розглянути наступний приклад: під час створення великої логістичної системи в Сингапурі спочатку використовувалася каскадна модель розробки, та проєкт зіткнувся зі значними проблемами під час виконання. Джефф де Лука вирішив переглянути підходи, застосувавши принципи ітеративно-інкрементної моделі. Цей новий підхід, що отримав назву “Feature Driven Development” (розробка, яка спрямована на функціонал), був заснований компанією Easel Corporation і призвів до створення методології SCRUM.

Шведський інженер А. Якобсон застосовував у своїй роботі ітеративний підхід і відмічав, що можливість до змін пріоритетів є гарною ідеєю, адже завдяки ітераціям можна поступово досягати покращень у функціональності.

У 1986 році американський інженер Баррі Боєм відкрив світу спіральну модель, котра була представлена методологією для розробки програмного забезпечення керування ризиками у процесі розробки. Відомо, що модель має в складових ітеративний та інкрементальний підхід.

Нижче наведена таблиця переваг та недоліків спіральної моделі.

Таблиця 1.3 – Переваги і недоліки спіральної моделі

Переваги спіральної моделі	Недоліки спіральної моделі
Перевагою є можливість в управлінні ризиками уже на ранніх етапах	До недоліків відносять складну реалізацію управління
До первеаг відносять гнучкість у зміні вимог	Недоліком є швидкі витрати бюджету
Системна інтеграція та валідація	При використанні спіральної моделі йде більше часу
Забезпечення реалістичних очікувань	Команда повинна бути на належному рівні (не підійде для розробників, що тільки починають свій шлях)
	Не підходить для малих проєктів

Дана модель має на меті змінімізувати ризики на початку ітерацій через посилення комунікації всередині команди.

На початку кожної ітерації спіральної моделі розробники визначають:

– цілі програмного фрагмента, розробленого під час цієї ітерації;

– основні та альтернативні шляхи досягнення цілей;

– аналіз обмежень.

Наступним етапом у роботі команди є виявлення проблем, які можуть виявити під час ітерації, а ще причин їх виникнення, що виникає через брак інформації чи недостатню кваліфікацію співробітників. Потім команда проєкту розпочинає роботу над стратегією реалізації проєктних робіт у межах цієї ітерації, враховуючи виявлені ризики. Кожен «спіральний виток» представляє повну частину продукту, що розробляється; отже, із кожним «поворотом» проєкт досягає глибшого рівня деталізації та витонченості.

Тож вибір моделі життєвого циклу необхідно обирати згідно з вимогами та специфікою проєкту. Також варто відмитити, що проєкт характеризується залежністю між трьома показниками: бюджетом (орієнтовною вартістю проєкту, закладеною в плані), терміном (часом, необхідним для виконання проєкту) і змістом робіт. Головним завданням є те, що потрібно знайти баланс між цими трьома частинами, щоб задовольнити вимоги та очікування клієнта, забезпечивши успішне завершення створення програмного продукту. Тому потрібен баланс між цими показниками. Стів Макконел зауважував, що створення програмного продукту на часі є складним процесом, бо він складається з багатьох етапів. Для досягнення продутивної команди необхідно застосовувати різні інструменти, які є направленими на стоврення ПЗ.

1.2 Аналіз методів гнучкої розробки

Із розвитком теорії управління розробкою програмного забезпечення проєктів фахівці цієї галузі зрозуміли, що комбінована версія ітераційної та спіральної моделей життєвого циклу може бути найкращим способом

організації процесу управління ІТ-проєктами. Поєднується поетапна розробка та наявність повних фрагментів програмного продукту, що працює від ітераційної моделі, включаючи домінуючу роль людського фактору та аналіз можливих ризиків ,узятих від спіральної моделі, еволюціонувала в нову методологію під назвою «гнучка» методологія(підходів до розробки ПЗ, що акцентують увагу на постійній комунікації та взаємодії всередині самоорганізованих груп фахівців).

Гнучкі методи направлені на удосконалення процесу розробки програмного продукту. Основною ідеєю є створення більш адаптивного та гнучкого підходу до розробки, який дозволяє краще реагувати на зміни та потреби користувачів.

Гнучкі методи (Agile) — це інноваційний підхід до розробки ПЗ ІС, спрямований на те, щоб зробити процес розробки більш гнучким і пристосованим до змін. Гнучкі методи (Agile) базуються на співпраці між клієнтами та розробниками, а також активній взаємодії з користувачами. Його основними цінностями є відкритість до змін, тісна комунікація, робота з живим програмним продуктом і готовність адаптуватися.

Гнучкі методи (Agile) використовують різноманітні методи та практики, включаючи систематичне тестування, вдосконалення коду та активне визначення вимог. Основними методами є Scrum, Kanban, Extreme Programming (XP). Вони спрямовані на оптимізацію процесу розробки, ефективне використання ресурсів та забезпечення високої якості продукту. Його можна успішно застосовувати в різних галузях, щоб задовольнити потреби клієнтів і користувачів, а не лише в розробці ПЗ.

Іван Григоренко у своїй книзі «Agile: Основи Agile-розробки» стверджує, що Agile – це більше, ніж просто методологія розробки ефективного кінцевого програмного продукту. Це своєрідний філософський

підхід, який відображається в культурі та цінностях команди. Він відзначається готовністю до змін у будь-якому етапі розробки. Відкритість для змін дозволяє ефективно реагувати на нові вимоги та обставини.

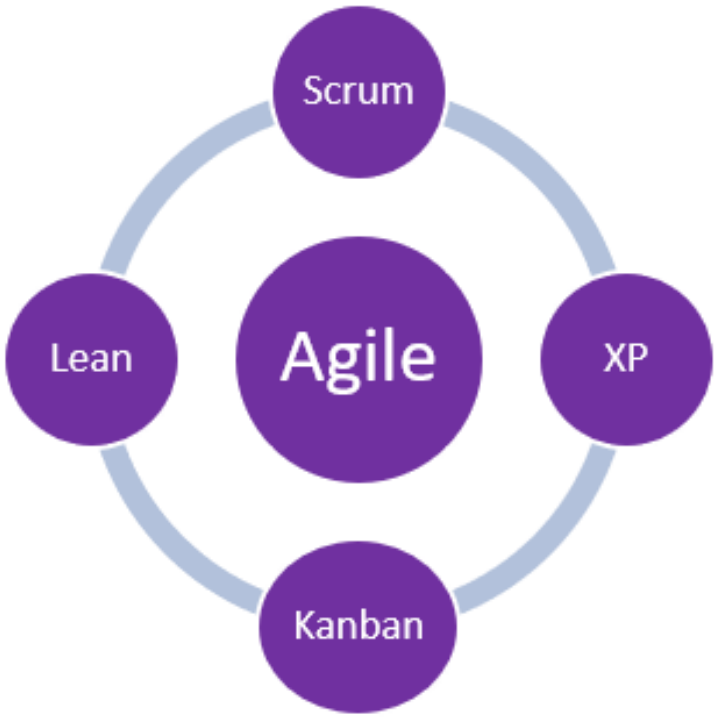

Рисунок 1.4 – Основні методи, що використовує Agile

PMI Ukraine (2020 р.) провели дослідження із застосування методологій компаній, що знаходяться в Україні. На базисі цього стало відомо, що 79% проєктів мали при створенні застосування гнучких методололгій (Agile).

Алістер Кокберн зазначав, що головною задачею є уміння довіритися своїй здатності у реагуванні на різноманітні події, аніж займатися плануванням. Тому пізніше був написаний «Маніфест Agile-розробки».

Маніфест Agile-розробки:

– важливішою є роль людей та їх взаємодія, ніж дотримання конкретних процесів і використання інструментів;

– продукт, що працює є важливішим, аніж обширна документація;

– співпраця і взаєморозуміння із замовником важливіші, аніж докладне оформлення умов контракту;

– готовність до змін важливіша, ніж точне виконання початкового плану.

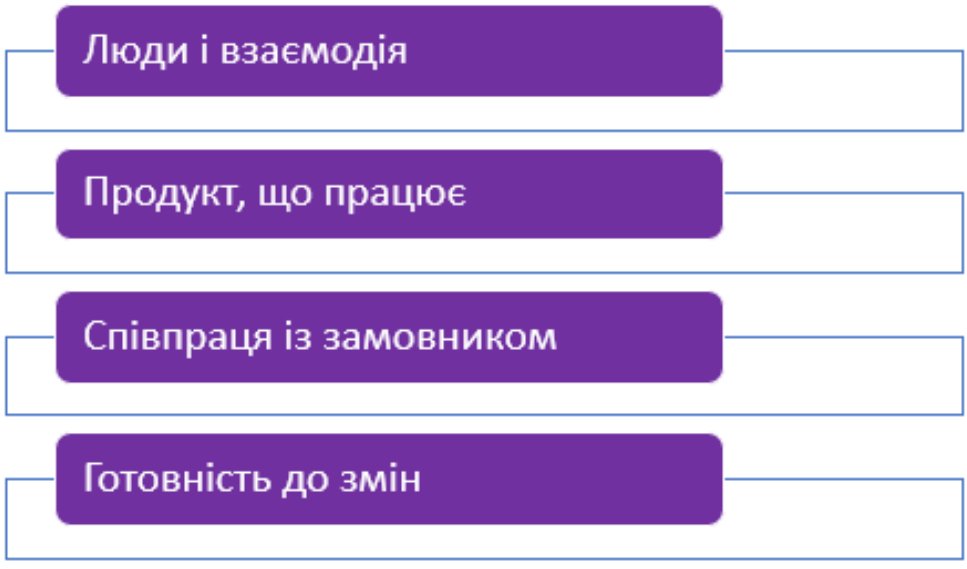

Рисунок 1.5 – Цілі маніфесту Agile

Далі розглянуто основні методи Agile:

– Kanban;

– Extreme programming (XP);

– Lean;

– Scrum.

Нижче наведено переваги і недоліки гнучких методів.

Таблиця 1.4 – Переваги і недоліки гнучких методів

Гнучкі методи	Переваги	Недоліки
Kanban	–зосередженість на поточних завданнях; –планове управління створенням ПЗ; –покращення продуктивності.	– проблемою є відсутність структурованих ітерацій; – ще однією проблемою виступає відсутність чітких термінів; – не підходить для роботи з чіткими термінами на виконання; – не підходить для багатозадачних проєктів.
Extreme Programming (XP)	–висока якість коду та ПЗ; –покращення якості комунікації команди; –швидка змінність.	–складність у дотриманні застосування коротких ітерацій та змінністю до вимог, що призводить до збільшення кількості завдань та відповідно навантажень на кожного члена команди, що призводить до змін у командній роботі. При великому навантаженні може бути серйозним викликом; –складність може торкнутися і питання часу, бо в методології є рекомендації з дотримання коротких ітерацій, але через велике наватаження на команду можливе недотримання цього графіка; –для малих команд може бути важко і непотрібно дотримуватися усіх практик і процедур, що призводить до збільшення складності; –виникає проблема, що стосується відповідальності. Адже на команді лежить відповідальність за якість розроблюваного продукту.
Lean	–мінімізація витрат; –покращення концентрації на створенні цінності для клієнта.	–недолік полягає у плануванні, на яке йде велика кількість часу; –може бути важко впровадженою у командах, де багато задач, адже тоді буде важко управляти розробкою програмного продукту; –не підходить тим, у кого вискоий контроль бюджету та часу.
Scrum	–чітке розподілення завдань та обов’язків; –прозорість у процесі розробки.	–часті зустрічі; –підходить лише для невеликих та середніх проєктів; –проблемою є визначення розміру команди; –результат залежить не лише від команди, а і від Scrum-майстра; –мінусом є потреба у регулярному спілкуванні із замовником (через відсутність зворотного зв’язку не виходить продовжувати роботу далі).

Kanban був розроблений японським інженером Taiichi Ohno в 1940-х роках. Інженер зрозумів, що виробничу систему Toyota можна вдосконалити. Структура Kanban перевела виробничий процес Toyota від процесу, коли продукт просувається на ринок, до процесу, де продукт створюється на основі ринкового попиту. Kanban є популярною гнучкою методологією, мета якої полягає в рівномірному розподілі навантаження між усіма учасниками проєкту, а також у намаганні зробити процес розробки прозорішим. Візуалізація усієї інформації проєкту, що дозволить краще знаходити помилки та швидко їх усувати; команда працює над одним завданням (буде усунено нерівноміірне навантаження та покращено результати); оптимізація процесу, строгий контроль часу на виконання задач – це три основних аспекти, яким потрібно слідувати для успішного використання даного підходу.

Розвиток програмного забезпечення інформаційних систем потребує змін, бо часто так буває, що успіх компанії залежить від швидкості, із якою вона може впроваджувати реалізацію продуктів на сучасному ринку. Із цього випливає, що потрібне скорочення життєвого циклу розробки ПЗ для компаній, де впроваджена розробка програмного забезпечення. Над цим і працював Кент Бек, створюючи гнучку методологію управління проєктами Extreme Programming (XP) (гнучка методологія програмного забезпечення інформаційних систем, в основі якої є набір принципів та практик, спрямованих на ефективне покращення якості продукту та поліпшення процесу розробки в цілому). Можна виділити 12 принципів, на яких заснована ця гнучка методологія: планування процесу, взаємодія із замовником, загальносистемні правила іменування класів та змінних, простота архітектури, рефакторинг, парне програмування, необхідність понаднормової роботи, єдині стандарти кодування, невеликі часті релізи, безперервна інтеграція й тестування. Мета полягає в адаптуванні програмного продукту до вимог, котрі увесь час змінюються та у підвищенні якості розробки. Варто відзначити, що постійна взаємодія поміж учасниками команди сприяє поліпшенню комунікації та кращому розумінню завдань. Хоча ця методологія є потужним

підходом, та все ж має певні недоліки, які відображені на таблиці вище.

Lean – це гнучка методологія, яка базується на концепції ощадливого виробництва. Принципи Lean основані на постійній роботі по скороченню витрат і залученню всіх працівників до процесу модернізації виробництва та максимальної уваги до клієнта. Сьогодні Lean є однією із найефективніших моделей формування майбутньої модернізації компанії, що використовує у своїй практиці такі принципи : формування потоку зі створення цінностей , надання стійкості потоку з формування цінностей, вилучення продукції для споживання. Це та тенденція вдосконалення , котра надає можливість компанії ефективно збільшувати трудову продуктивність, можливості до конкуренції і якості, не потребує великих витрат на модифікацію. Вона схожа на Scrum тим, що загальна задача розбивається на менші завдання, і кожне з них реалізовується окремо. Розписується чіткий порядок послідовностей , який не можна порушувати. Однією з головних переваг є надання акценту на розвиток оптимізації процесів, уникнення зайвого та виділення цінностей для клієнта. Тут може виникнути проблема для малих команад із великою кількістю завдань та високим темпом роботи у важкості витриманні балансу (між процесом і роботою над функціоналом). Перевагаю є той факт, що Lean вважається гнучким підходом, адже принципи цього методу формуються з потреб команди.

Основою Scrum є спринт, під час якого буде виконана робота над продуктом. Наприкінці спринту необхідно отримати нову робочу версію продукту. Важливо зазначити , що спринти завжди мають обмеження в часі (1-4 тижні) та однакову тривалість протягом усього терміну «життя» продукту. Планування відбувається на початку кожного спринту , а оцінка змісту беклога спринту – це ті завдання, що виконують у початковому спринті. Кожен спринт повинен мати мотивуючий фактор – мету, а мета спринту досягається шляхом

виконання завдань беклогу. Проводяться щоденні зустрічі, на яких кожен член команди запитує та надає відповіді на питання: «Що ти зробив вчора?» і «Що плануєш робити сьогодні?». Мета такої зустрічі – оцінити стан і прогрес спринту, виявити перешкоди на ранніх стадіях і сформулювати рішення для зміни стратегій, необхідних для досягнення цілей спринту. Далі проводяться рев'ю і ретроспектива, їх завдання – оцінка ефективності (продуктивності) команди в минулому спринті, прогнозування очікуваної ефективності (продуктивності) в наступному спринті, пошук проблем, що виникали , оцінка вірогідності завершення усіх необхідних робіт щодо продукту та інше. Рекомендаційний розмір команди – 7 осіб. Згідно з ідеологією Scrum, команди більшого складу вимагають надто великих ресурсів на комунікації, тоді як команди меншого складу можуть зустрітися з підвищенням ризиків (за рахунок можливої відсутності необхідних навичок) та зменшують розмір роботи, котрий команда може виконати за одиницю часу.

Scrum складається із трьох головних компонентів:

– ролей;

– артефактів;

– процесів.

Динаміку популярності пошукових запитів методів гнучкої розробки на території України, представленої сервісом GoogleTrends, показано на рисунках нижче.

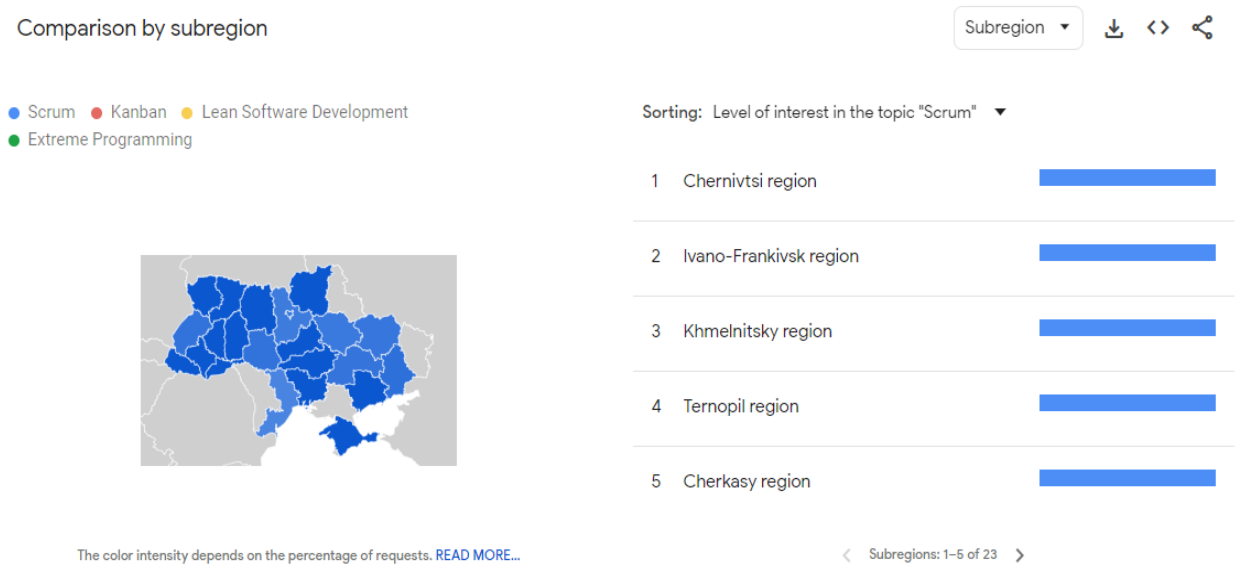

Рисунок 1.6 – Аналіз запиту з питання популярності методології Scrum

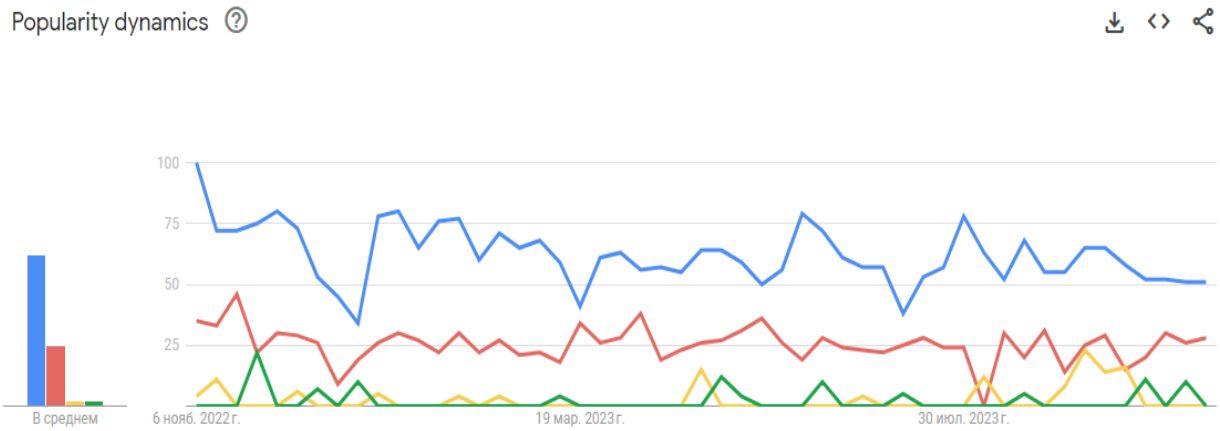

Рисунок 1.7 – Динаміка популярності за останній рік

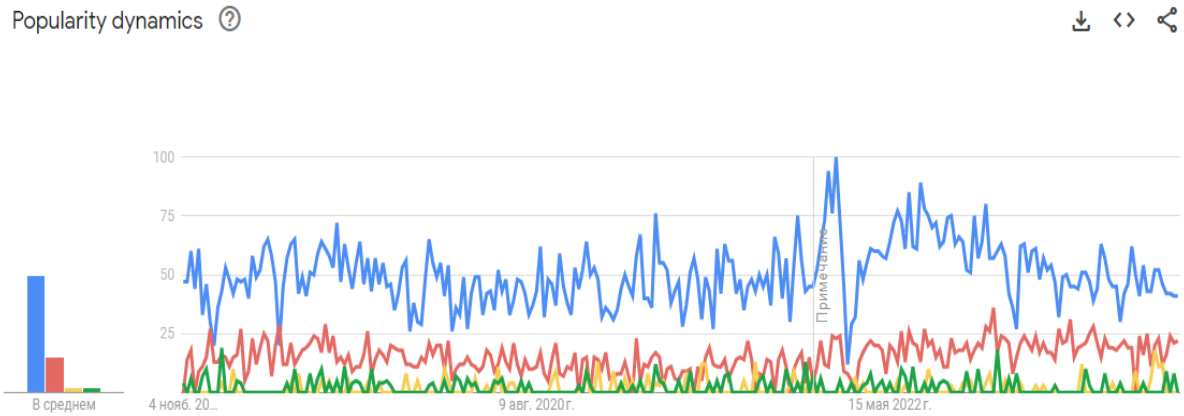

Рисунок 1.8 – Динаміка популярності за останні 5 років

Із цієї динаміки видно, що найбільшої популярності в Україні за останній рік зазнали Scrum і Kanban.

Дані методи можуть перетинатися або комбінуватися завдяки гнучкості. Наприклад, представлення завдань у вигляді карток, розташованих на дошці Kanban, що вказує на статус завдання, є поширеним і в інших методологіях. Спринти (ітерації розробки) згадуються у більшості методологій.

Із аналізу, котрий подано на рисунку вище , видно, що інтерес до Scrum найбільше проявляють мешканці Чернівецької, Івано-Франківської, Хмельницької, Тернопільської та Черкаської областей. Scrum став популярним у світі та широко застосовується у сфері інформаційних технологій. Якщо врахувати розвиток цієї галузі в Україні, можна відзначити, що знання Scrum є важливою конкурентною перевагою для IT-спеціалістів.

1.3 Дослідження підходів до побудови процесних моделей

У сьогоденні процесні методи допомагають у аналізі та удосконаленні бізнес-процесів, де за основу виступає інформація про структуру та поведінку процесів, узята із логів (ще називають журналом подій), де збережено дані про зміни станів об’єктів.

Уперше про управління бізнес-процесами були загадано у 1980 роках. Взагалі поняття бізнес-процес виступає як потік послідовних дій, після завершення яких, можна отримати «на виході» програмний продукт. Цикл управління бізнес-процесом налічує такі етапи: визначення процесу, його аналізу, реалізації змін, моніторингу процесу та оптимізації процесу.

Рисунок 1.9 – Цикл управління бізнес-процесом

Авторами перших робіт в області процесних методів були Джонатан Кук і Олександр Вольф. Аналіз процесних методів й надалі допомагає у наданні інформації, що стосується наскрізних процесів. Тому процесні методи активно застосовують у більшості етапів життєвого циклу. Процесні методи можуть бути корисними у виявленні проблем, у тому, щоб зробити рентабельні дії пріоритетними, у збереженні часу проходження, у підвищенні задоволеності клієнтів, а також у запуску чи масштабуванні автоматизації процесів. Варто відмітити, що процесні методи також допоможуть у структуруванні та визначенні вузьких місць у будь-якому типі процесів. Також потрібно приділяти достатньо уваги логам, адже якщо вони будуть неповними, то і модель стане недостовірною. А це несе за собою проблеми, пов'язані із аналізом, неточність характеристик процесу, що може призвести до хибного розуміння реального стану процесу. Логи є послідовністю дій, котрі виконуються в системі чи процесі, тобто допомагають у фіксації дій, що виконуються в ході роботи процесу.

Нідерландський учений Ван Дер Аалст, який досліджував інформаційні технології та бізнес-процеси, зазначав, що логи є джерелом персоналізованих даних в аналізі та оптимізації бізнес-процесів.

Автоматизацію називають важливою умовою підвищення ефективності процесів, її перевага – це змога накопичувати значні обсяги даних, а виявлені результати допомагають в отриманні достовірної (повної) картини процесів, що може відрізнятися від передбачуваної.

Процесні методи сприяють оперативному прийняттю оптимальних рішень та отриманню максимуму вигоди із наявних можливостей, які є важливими в даний час для усіх компаній бізнес-цілями, котрі не залежать від процесів компанії.

До задач, у вирішенні яких можуть допомогти процесні методи, відносять:

– вилучення;

– зіставлення (перевірка відповідності реального екземпляра процесу базовому);

– покращення процесів.

Вилучення процесу несе за собою побудову моделі процесу за даними із журналу подій інформаційної системи, що працює. Перевірка щодо відповідності реального екземпляра процесу базовому дозволяє виявити відхилення між моделлю, яка існує, та логом подій. Покращення, у свою чергу, дозволяє поліпшити модель процесу за рахунок розширення поведінки, що моделюється, і отримання великої моделі показників процесу.

Рисунок 1.9 – Застосування процесних методів

Для розпізнавання та аналізу процесів використовуються такі алгоритми:

– альфа-алгоритм;

– евристичний алгоритм;

– генетичний алгоритм;

– індуктивний алгоритм;

– нестрогий алгоритм (ще зустрічається як «Fuzzy»).

Альфа-алгоритм був створений Ван Дер Аалстом у 2013 році. Вважається, що саме цей алгоритм є одним із перших та найпростіших алгоритмів вилучення моделі процесу. Основною перевагою є той факт, що алгоритм спрямований на максимізацію критерію простоти моделі. Важливою деталлю є те, що протокол повинен бути без шуму, а також повинен бути доповненим відношенням потоків. Даний алгоритм варто застосовувати, щоб виявити паралельну активність, адже, застосовуючи подвійні позначки часу, альфа-алгоритм виявляє події початку й кінця.

Евристичний алгоритм базується на тому, що модель, яку отримуємо за допомогою нього, має інформацію про використання вузлів і дуг у той час, коли виконуються логи подій. Однією з відмінностей евристичного алгоритму є те, що він зважає тільки на порядок подій у процесі. Ще алгоритм допомагає у питанні, зв'язаному із шумами, а також видає модель, що показує тільки основну поведінку із записаного в логах подій, бо деталі процесу відкидаються даним алгоритмом. Евристичний алгоритм складається із наступних етапів: побудови графу залежності, вхідних та вихідних виразів для кожної діяльності, а також пошуку залежностей взаємозв'язків на великій відстані.

Генетичний алгоритм спрямований на виявлення задач, що дублюються. Цей алгоритм складається з наступних етапів: побудови сукупності моделей процесу, де до кожної моделі зіставляється функція, котра

дає оцінку, наскільки добре модель відтворює поведінку в лозі. За допомогою операторів перетину та мутації відбувається покращення моделей, котрі називають у даному алгоритмі «еволюцією». На відміну від інших алгоритмів процесних методів, на виконання генетичного алгоритму йде велика кількість часу.

Індуктивний алгоритм у порівнянні з іншими є наймолодшим. Цей алгоритм дозволяє виявити «розвилки» між початковою та кінцевою стадіями процесу. Він розглядає логи подій з вірогідної сторони, що дозволяє мати коректний кінцевий результат (модель), що працює. Перевагою індуктивного алгоритму є те, що він може взаємодіяти із неповними логами подій.

Нестрогий алгоритм містить елементи нечіткості й потребує гнучкості у моделюванні. Він надає можливість для подальшого аналізу та оптимізації процесів. Енн Розіант у своїй праці «Процес інтелектуального аналізу: відповідність та розширення» наголошує на тому, що нестрогий алгоритм використовують на практиці для точного виявлення моделей процесу й візуалізації складних процесів.

Під час оцінювання моделі необхідно звернути увагу на такі метрики якості процесу:

– придатність (оцінка моделі залежно від її зіставлення з екземплярами процесу);

– точність (відображення численних варіантів виконання процесу);

– простота (досягається завдяки зниженню рівня придатності й точності);

– узагальненість.

1.4 Постановка задачі

Дана робота направлена на дослідження процесних методів гнучкої розробки програмного забезпечення інформаційних систем. Ця задача на сьогоднішній день актуальна, адже завдяки удосконаленню методу гнучкої розробки програмного забезпечення відбувається зниження ризиків і значно підвищується продуктивність команди.

Метою роботи є дослідження процесних методів гнучкої розробки програмного забезпечення інформаційних систем.

Для досягнення мети поставлено такі задачі:

– розгляд інструментів, які існують для поліпшення процесу розробки програмного забезпечення інформаційних систем;

– вибір найбільш ефективних інструментів;

– проведення аналізу процесу розробки програмного забезпечення;

– проведення аналізу гнучких методів Agile;

– проведення аналізу процесних методів, що існують;

– встановлення найефективнішого шляху вирішення проблеми;

– побудова оптимальних процесів управління розробкою програмного забезпечення інформаційних систем;

– проведення експериментальної перевірки.

2 ОЦІНКА РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНИХ СИСТЕМ ІЗ ВИКОРИСТАННЯМ ТЕМПОРАЛЬНИХ ЗАЛЕЖНОСТЕЙ

2.1 Темпоральний опис процесу розробки програмного забезпечення інформаційних систем

Під визначенням методологія Scrum розуміють згуртовану роботу усієї команди, у тому числі розробників і тестувальників. Завдання, котрі будуть опрацьовані командою розробки, втілюють бачення менеджера проєктів та аналітика. Результатом спринту виступає готовий програмний продукт або його частина, залежно від поставлених задач.

Процес виконання розробки програмного забезпечення інформаційних систем складається з наступних етапів:

– Product Backlog (на цьому етапі варто розглянути перелік задач, котрі будуть розроблені, даючи їм оцінку важливості, щоб полегшити роботу команди);

– Sprint Planning (даний етап має на меті обрання переліку задач із попереднього етапу, котрі будуть виконуватись у даному спринті, зважаючи на оцінку важливості, User Stories);

– Sprint Backlog (даний етап складається із результатів етапів Product Backlog і Sprint Planning, що має бути деталізованим задля розгляду прогресу командою розробників під час щоденних зустрічей);

– Sprint Execution (на цьому етапі відбувається виконання поставлених задач командою);

– Daily Scrum (щоденна зустріч, що складається з 15 хвилин, котра має на меті синхронізацію, створення плану на конкретний день);

– Sprint Review (даний етап має на меті отримання інкременту продукту, отримання зворотного зв’язку, відбувається 1 раз під кінець спринту);

– Sprint Retrospective (метою виступає аналіз роботи, планування покращень, на даному етапі відбувається розгляд вузьких місць роботи команди);

– Potentially Shippable Increment (додатковий етап, який вказує на можливість забезпечення інкременту приросту, котрий можливо одразу впровадити).

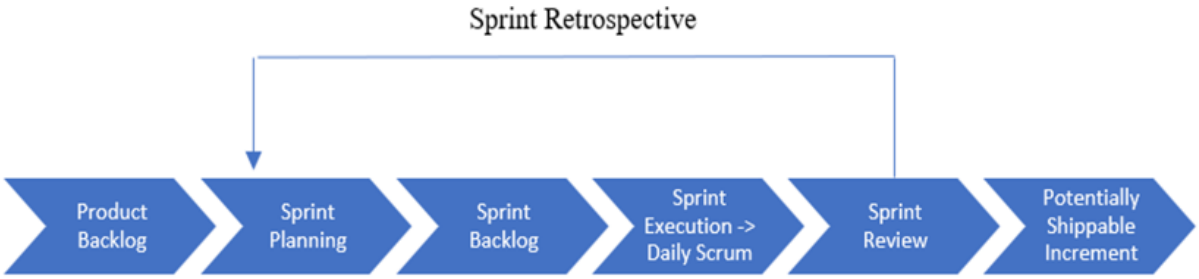

Рисунок 2.1 – Загальна послідовність етапів циклу гнучкої розробки ПЗ ІС із методологією Scrum

На етапі Sprint Retrospective відбувається надання оцінки роботи команди, розглядається оцінка складності задач, відстежуються вузькі місця розробки й планування реалізації розробки програмного забезпечення в цілому.

Sprint Retrospective складається з таких етапів: підготовки, збору даних, генерування ідей, планування дій у подальшому та кінця ретроспективи. Дані

етапи надають змогу ефективно використовувати їх результат у подальшій роботі.

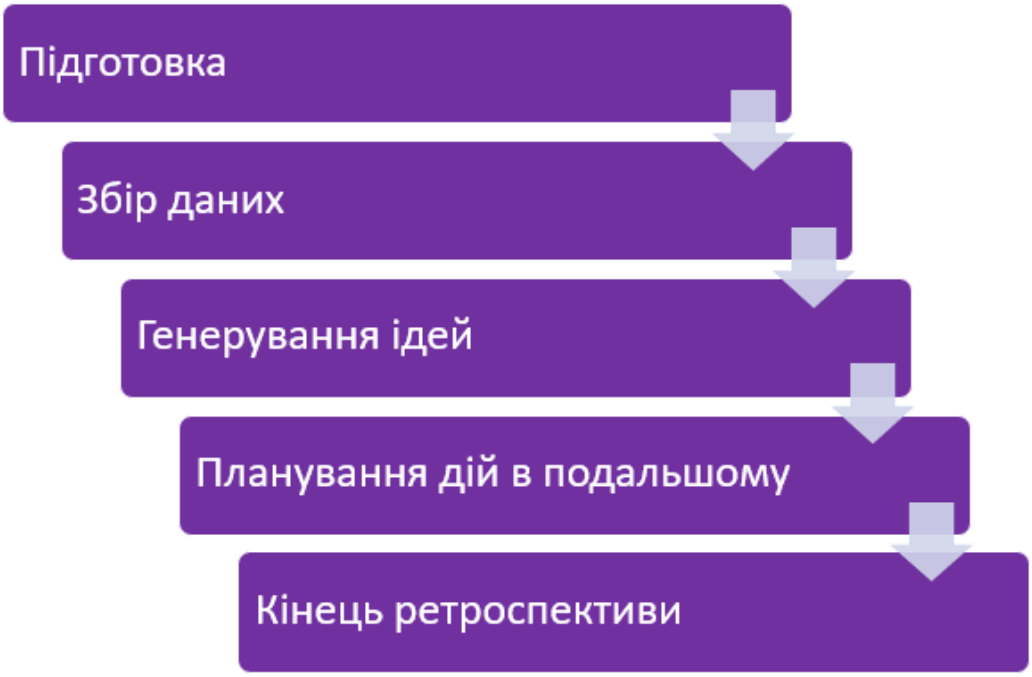

Рисунок 2.2 – Послідовність етапів Sprint Retrospective

Під час проведення даного етапу циклу гнучкої розробки програмного забезпечення ІС проводиться обробка великої кількості інформації, де однією із основних складових є час. А його невикористання несе за собою неналежне сприйняття інформації.

Варто зазначити, що характерним для цього етапу є виявлення неодноразового виникнення затримок (вузьких місць), які наразі

з’ясовувалися по пам’яті членами команди розробки, що не можна назвати ефективним способом.

Під час дослідження процесних методів у попередньому розділі було отримано такі алгоритми:

– альфа-алгоритм;
– евристичний алгоритм;
– генетичний алгоритм;
– індуктивний алгоритм;
– нестрогий алгоритм.

Базуючись даними, які було отримано, розроблено удосконалений метод. Наявні алгоритми орієнтовані на аналіз та покращення процесу розробки програмного забезпечення інформаційних систем. У даній роботі використано нестрогий алгоритм, адже він добре працює з логами подій. Логи подій є операціями у вигляді протоколів, що допомагають в аналізі процесу. Вони дозволяють розглянути не тільки процес, але і його складові.

Логи подій зазвичай мають 4-5 атрибутів:

– Case Id (ідентифікатор подій);
– Activity (подія);
– Timestamp – Start Time (час початку події);
– Timestamp – Completion Time (час завершення події);
– Resource (виконавець події).

Наразі удосконалення етапу ретроспективи процесними методами за допомогою темпоральних правил є актуальним на сьогоднішній день. Завдяки йому відбудеться ефективний аналіз інформаційних систем із розробки програмного забезпечення.

Варто зазначити, що темпоральні залежності зв’язані з конкретними відрізками часу. У темпоральних правилах факти представлені як блоки даних із позначкою часу, що використовується задля визначення періоду часу.

Метод, котрий розглядає етап ретроспективи методології Scrum, що базується на темпоральних правилах.

Етап 1. Побудова темпоральних правил.

Етап 2. Визначення кванторів до правил.

Етап 3. Розрахунок часу виконання

Етап 4. Перевірка затримок по правилам

Етап 5. Визначення активності та виконавців із затримок.

Темпоральне правило $n_{\mathrm{m}}^{\mathrm{j}}$ типу «neXt»:

$$\mathrm{n}_{\mathrm{m}}^{\mathrm{j}} \equiv F_{\mathrm{j}} \overset{N}{\rightarrow} \mathrm{F}_{\mathrm{m}}\,, \qquad (2.1)$$

де $\mathrm{F_j}$ і $\mathrm{F_m}$ – послідовні факти (початок і кінець розробки);

N – темпоральний оператор.

Із цього маємо темпоральну упорядкованість:

$$\left(\forall \mathrm{l_{i,q}} \in \Pi_{\mathrm{i}}\right)\exists\left(\mathrm{l_{i,j}}, \mathrm{l_{i,m}}\right)\colon \left(\mathrm{t_{i,q}} < \mathrm{t_{i,j}}\right)\vee\left(\mathrm{t_{j,m}} < \mathrm{t_{i,q}}\right) \Rightarrow \exists \mathrm{n_m^j}, \qquad (2.2)$$

де $\mathrm{l_j}$, $\mathrm{l_m}$– факти виникнення стану процесів;

Π_{i} – послідовність;

$\mathrm{t_{i,j}}$, $\mathrm{t_{i,m}}$ – початок і кінець події;

$\mathrm{t_{i,q}}$ – неможливість проміжків між початоком та кінцем події;

$l_{i,q}$– стан об’єкту управління.

Темпоральне правило типу «neXt» з квантором E:

$$En_m^j \equiv F_j \overset{EN}{\rightarrow} F_m | (\exists V_i): F_{i,j} \overset{N}{\rightarrow} F_{i,m} \ , \qquad (2.3)$$

Темпоральне правило типу «neXt» з квантором A:

$$An_m^j \equiv F_j \overset{AN}{\rightarrow} F_m | (\forall V_i) F_{i,j} \overset{N}{\rightarrow} F_{i,m}, \qquad (2.4)$$

Темпоральне правило u_m^j типу «Future»:

$$u_m^j \equiv F_j \overset{U}{\rightarrow} F_m \ , \qquad (2.5)$$

де F_j і F_m – послідовні факти (початок і кінець розробки);

U – темпоральний оператор.

Із цього маємо темпоральну упорядкованість:

$$(\forall l_{i,q} \in \Pi_i) \exists (l_{i,j}, l_{i,m}): t_{i,j} < t_{i,q} < t_{i,m}, \qquad (2.6)$$

де l_j , l_m– факти виникнення стану процесів;

Π_i – послідовність;

$t_{i,j}$, $t_{i,m}$ – початок і кінець події;

$t_{i,q}$ – неможливість проміжків між початоком та кінцем події;

l_q– стан об’єкту управління.

Темпоральне правило типу «Future» з квантором E і квантором A:

$$Eu_m^j \equiv F_j \overset{EU}{\rightarrow} F_m | (\exists V_i): F_{i,j} \overset{U}{\rightarrow} F_{i,m}\ , \tag{2.7}$$

$$Au_m^j \equiv F_j \overset{AU}{\rightarrow} F_m | (\forall V_i) F_{i,j} \overset{U}{\rightarrow} F_{i,m}, \tag{2.8}$$

2.2 Метод побудови темпоральної моделі процесу розробки програмного забезпечення

Процесні методи допомагають у аналізі показників часу за допомогою наглядної картини, що надає побудова моделей процесу.

Далі відбувається порівняння між ідеальними та реальними строками виконання процесу:

$$\Delta t = t_c - t_b\ , \tag{2.9}$$

де t_c – реальний;

t_b – ідеальний;

Δt – різниця часових показників.

Одразу після цього визначається за допомгою кванторів A та E варіанти проблематики даних, де A – події виконується за одним сценарієм, а E – події можуть виконуватись за одним сценарієм, а можуть і за іншим.

3 ТЕХНОЛОГІЯ ПОБУДОВИ МОДЕЛІ І ОЦІНКИ ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІЗ ВИКОРИСТАННЯМ ТЕМПОРАЛЬНИХ ЗАЛЕЖНОСТЕЙ

3.1 Опис інформаційної технології

Даний розділ має на меті показати етапи реалізації й оптимізації процесів гнучкої розробки програмного забезпечення інформаційних систем.

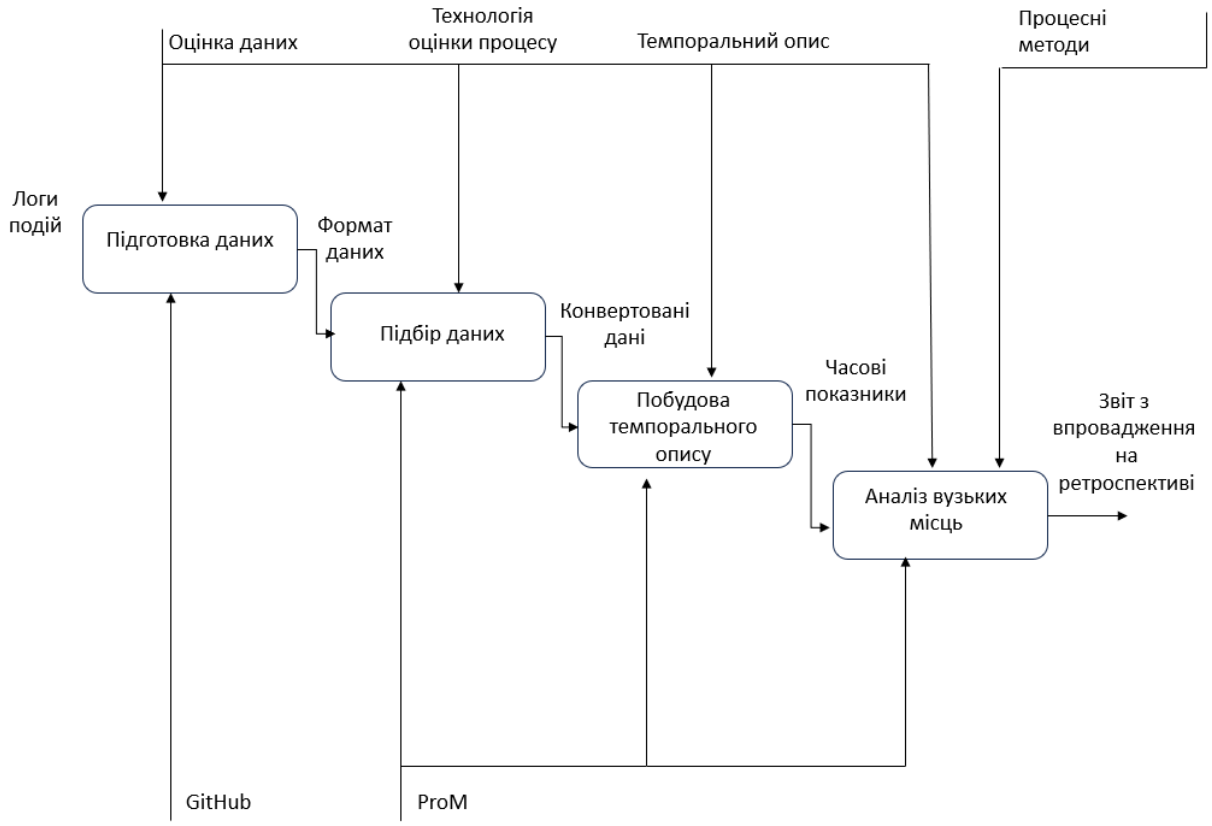

Рисунок 3.1 – Послідовність етапів аналізу процесів

Перший етап спрямований на проведення підготовки даних.

На другому етапі відбувається підбір даних. Далі відбувається побудова темпорального опису.

Заключним етапом є аналіз вузьких місць.

3.2 Імплементація інформаційної технології

Вхідною інформацією виступає документація про роботу команди над програмним продуктом, що може відобразити усі наявні зв'язки. Це представлено на рисунках 3.2 та 3.3. Даний проєкт налічує чотири розробника, тестувальника, менеджера проєктів, бізнес-аналітика та девопса. Використовується методологія Scrum. У даній гнучкій методології зазвичай відбувається зафіксування довжини спринта від одного до чотирьох тижнів. У даній ситуації було обрано довжину спринта в 2 тижні. Цей варіант є оптимальним, адже команда розробників має можливість планування задач під свої потреби та менше переживає через брак часу, як це могло бути зі спринтом в 1 тиждень; є можливість скорого обговорення із замовником невеликих змін по його бажанню; перевагою є той факт, що є можливість деталізації кожної деталі із усього переліку задач, що дозволяє у проєктуванні задачі; також звернено увагу на усі деталі, котрі найчастіше команда виявляє під кінець ітерації. Також варто звернути увагу на рівень важкості процесу розробки, що наведено на рисунку нижче під заголовком Estimation (під цим поняттям розуміють час, який розрахований, щоб отримати готовий варіант створення або редагування задачі).

SHA	Item	Issue Summary	Date	Resource	Type	Activity
e6124dc74b	RR-1112	Peradventure - tax fiel	2023-11-06 12:50:03	Developer 2	Developmen	join new logic with frontend validation
9b28088231	RR-1112	Peradventure - tax fiel	2023-11-03 12:40:12	Developer 1	Developmen	Discussion problem with infinite loading the UI for new table and fix it
59f35a59d8	RR-1112	Peradventure - tax fiel	2023-11-02 15:00:10	Developer 2	Developmen	Try to fix infinite loading the UI for new table
ea5dfe19f4	RR-1112	Peradventure - tax fiel	2023-11-07 12:38:01	Developer 1	Developmen	Update migration file
7ae24c8705	RR-1108	Peradventure - store cu	2023-11-06 13:50:00	Developer 2	Developmen	Passing and getting tax length
6db4705763	RR-1108	Peradventure -block_1	2023-11-10 12:30:02	Developer 2	Developmen	Update frontend to use only Tax Length
3c2a5981b6	RR-1108	Peradventure -block_1	2023-11-10 17:03:40	Developer 2	Developmen	Update label in agreement_user_identifier
2153e78005	RR-1108	Peradventure -block_1	2023-11-08 15:00:06	Developer 2	Developmen	Show warning message if user try to change ID length for location with employees
a08d9f0518	RR-1108	Peradventure -block_1	2023-11-08 18:02:40	Developer 2	Developmen	Change label in agreement_user_identifier by length
068deac28f	RR-1108	Peradventure -block_1	2023-11-10 17:34:11	Developer 2	Developmen	Add one more label to agreement_user_identifier
45e8c0519f	RR-1108	Peradventure -block_1	2023-11-09 13:09:26	Developer 2	Developmen	Pass has_employees for each location in getlocalsites
5ca8286dfc	RR-1115	Peradventure - store cu	2023-11-14 14:40:11	Developer 3	Developmen	Update backend to return addition data
80f54404b1	RR-1115	Peradventure - store cu	2023-11-01 15:00:00	Developer 2	Developmen	Update service to return tax limit for several locations and updated unit test
d3bbb62b4f	RR-1115	Peradventure - store cu	2023-11-01 18:20:00	Developer 2	Investigatior	RR-1115 webshop optimizations
adee810477	RR-1115	Peradventure - store cu	2023-11-09 14:13:01	Developer 2	Developmen	RR-1115 added type
9ed9f049d9	RR-1115	Peradventure - store cu	2023-11-13 13:40:00	Developer 1	Developmen	RR-1115 build
fb15574d4d	RR-1115	Peradventure - store cu	2023-11-07 17:35:00	Developer 1	Developmen	Branch 'develop' into RR-1115
9cd656c439	RR-1115	Peradventure - store cu	2023-11-15 12:50:11	Developer 2	Developmen	RR-1115 fix show
bfa4c509ab	RR-1115	Peradventure - store cu	2023-11-01 10:00:00	Developer 2	Developmen	RR-1115 Branch 'develop'
dedd97f532	RR-1115	Peradventure - store cu	2023-11-01 15:20:00	Developer 1	Investigatior	RR-1115 styling
b27bfac9d6	RR-1115	Peradventure - store cu	2023-11-01 20:35:00	Developer 1	Developmen	RR-1115 build
dcadecba5a	RR-1115	Peradventure - store cu	2023-11-01 13:20:00	Developer 3	Investigatior	RR-1115 styling
50a526dda8	RR-1115	Peradventure - store cu	2023-11-01 16:21:00	Developer 3	Investigatior	RR-1115 build
8ec4c07f7c	RR-1112	Peradventure - tax fiel	2023-11-01 20:35:00	Developer 4	Developmen	Branch 'develop' into RR-1112
0a3d9a1117	RR-1112	Peradventure - tax fiel	2023-11-01 16:35:00	Developer 4	Developmen	RR-1112 added parametr
9f9f3b58a3	RR-1112	Peradventure - tax fiel	2023-11-01 18:55:00	Developer 4	Investigatior	RR-1112 build
d6bf6a379d	RR-1112	Peradventure - tax fiel	2023-11-02 18:01:10	Developer 2	Developmen	RR-1112 fixed
6b434e0faf	RR-1112	Peradventure - tax fiel	2023-11-02 21:00:10	Developer 2	Developmen	RR-1112 adding handling events
0f9920da32	RR-1112	Peradventure - tax fiel	2023-11-02 11:00:10	Developer 1	Developmen	RR-1112 handing events (fixed)
961a5f15fa	RR-1112	Peradventure - tax fiel	2023-11-02 15:13:10	Developer 1	Developmen	RR-1112 adding handing events
9cb84f06d1	RR-1112	Peradventure - tax fiel	2023-11-02 13:01:10	Developer 3	Developmen	RR-1112 adding frontend logic to JS/React
0faceaa949	RR-1112	Peradventure - tax fiel	2023-11-02 15:47:10	Developer 3	Developmen	RR-1112 fixed

Рисунок 3.2 – Дані, що описують процес розробки програмного забезпечення ІС

Num	Item	Issue Summary	Date	Comment	Spent time	Estimation
4	RR-1108	Time logging meetings	2023-11-21 10:00:00	Team Weekly	1h 15m	
6	RR-1115	Time logging meetings	2023-11-14 10:00:00	Team Weekly	1h	
304	RR-1112	Peradventure - tax field len	2023-11-01 10:00:00	RR-1112 build	2h 30m	10h 30m
307	RR-1112	Peradventure - tax field len	2023-11-06 10:00:00	RR-1112 added type	1h	10h 30m
309	RR-1112	Peradventure - tax field len	2023-11-03 10:00:00	Fixed logic	1h	10h 30m
311	RR-1112	Peradventure - tax field len	2023-11-02 10:00:00	added new condition	3h	10h 30m
314	RR-1115	Peradventure - store custo	2023-11-20 15:00:00	Fix code by recomme	2h	10h 30m
315	RR-1112	Peradventure - tax field len	2023-11-07 10:00:00	Device online and se	2h 40m	10h 30m
316	RR-1112	Peradventure - tax field len	2023-11-06 12:00:00	webshop: Show reser	2h 30m	10h 30m
318	RR-1112	Peradventure - tax field len	2023-11-10 10:00:00	Update label in agree	30m	10h 30m
319	RR-1112	Peradventure - tax field len	2023-11-10 11:00:00	Show warning messa	2h 30m	10h 30m
320	RR-1112	Peradventure - tax field len	2023-11-08 10:00:00	Change label in agree	2h 30m	
321	RR-1112	Peradventure - tax field len	2023-11-08 13:00:00	Add one more label t	30m	
322	RR-1112	Peradventure - tax field len	2023-11-10 15:00:00	Pass has_employees	1h	
323	RR-1112	Peradventure - tax field len	2023-11-09 10:09:00	Update backend to re	1h 30m	
324	RR-1115	Peradventure - store custo	2023-11-24 10:10:00	adding frontend logic	2h	4h
325	RR-1115	Peradventure - store custo	2023-11-24 13:00:00	Update coffee script t	3h	4h
326	RR-1115	Peradventure - store custo	2023-11-16 10:00:00	Feature for the applic	3h	
327	RR-1115	Peradventure - store custo	2023-11-14 10:00:00	Research common lo	2h	
328	RR-1115	Peradventure - store custo	2023-11-17 10:02:00	adding new elements	2h	
329	RR-1108	Peradventure -block_1 sup	2023-11-21 10:10:00	Fix source code by re	2h 45m	
330	RR-1115	Peradventure - store custo	2023-11-20 10:02:00	adding backend for pr	1h	
331	RR-1112	Peradventure - tax field len	2023-11-07 15:00:00	Branch 'develop' into	2h	4h
332	RR-1112	Peradventure - tax field len	2023-11-07 18:00:00	Frontend service to g	1h	4h
333	RR-1112	Peradventure - tax field len	2023-11-09 15:00:00	Getting label name fo	2h	4h
334	RR-1112	Peradventure - tax field len	2023-11-13 10:01:00	Update frontend serv	3h	4h
352	RR-1108	Peradventure -block_1 sup	2023-11-21 10:02:10	Team Weekly	1h 15m	4h
358	RR-1112	Peradventure - tax field len	2023-11-07 10:20:00	Team Weekly	50m	
371	RR-1115	Peradventure - store custo	2023-11-14 10:00:00	Team Weekly	1h	
374	RR-1112	Peradventure -block_1 sup	2023-11-28 18:00:00	Team Weekly	1h	
768	RR-1112	Peradventure - tax field len	2023-11-02 16:20:00	debugging and fixing	3h	
769	RR-1112	Peradventure - tax field len	2023-11-07 10:10:02	styles improvement	2h	
770	RR-1112	Peradventure - tax field len	2023-11-07 13:00:00	added new logic for h	4h	
771	RR-1115	Peradventure - store custo	2023-11-15 10:00:00	Integration	2h	
772	RR-1115	Peradventure - store custo	2023-11-15 13:00:00	adding frontend logic	4h	
773	RR-1115	Peradventure - store custo	2023-11-17 10:00:00	styling	2h	
774	RR-1108	Peradventure -block_1 sup	2023-11-23 10:10:00	meeting	15m	
775	RR-1108	Peradventure -block_1 sup	2023-11-22 12:00:02	meeting	15m	
776	RR-1108	Peradventure -block_1 sup	2023-11-22 14:00:34	changing backend log	2h	
777	RR-1108	Peradventure -block_1 sup	2023-11-22 11:00:24	creating and local tes	2h	
778	RR-1108	Peradventure -block_1 sup	2023-11-23 12:00:00	debugging, building, r	1h 30m	
779	RR-1108	Peradventure -block_1 sup	2023-11-23 13:00:00	changing reports	4h	
780	RR-1108	Peradventure -block_1 sup	2023-11-23 15:00:00	changing column nam	1h	
781	RR-1108	Peradventure -block_1 sup	2023-11-22 15:00:00	adding new frontend	3h 30m	
790	RR-1108	Peradventure -block_1 sup	2023-11-27 10:00:20	fixed logic	2h	

Рисунок 3.3 – Дані, що описують процес розробки програмного забезпечення ІС одного розробника

4 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ЗАПРОПОНОВАНИХ ТЕОРЕТИЧНИХ РЕЗУЛЬТАТІВ

4.1 Опис платформи для експериментальної перевірки

Усе частіше для візуалізації бізнес-процесів використовують процесні методи. Вони допомагають побудувати наглядні моделі, що будуть відображати специфіку процесів із логів подій.

Процесні методи допомагають вирішити проблеми, що стосуються наступних аспектів: наявність складних структур у логах подій; складність самої моделі, адже аналітику важко сприймати діаграми, де є велика кількість переходів або подій, що повторюються; проблема може виникнути через неповноту логів подй.

У даній роботі було вирішено використовувати інструмент ProM. Він створений для інтелектуального аналізу процесу такими розробниками: Вілл ван дер Аалстом, Пітером ван дер Брандом, Массіміліано де Леоні, Будевійн ван Дорген, Дірк Фаландем та іншими. ProM використовують для аналізу журналів подій (логів), котрі направлені на завантаження файлів із даними, обробки та фільтрації. Даний інструмент надає можливість у візуалізації процесів, а ще допомагає виявити їх закономірності та проблемні місця. Цей інструмент має відкритий вихідний код. ProM містить у собі можливість використання алгоритмів і типів моделей процесів для ефективного аналізу логів подій.

До переваг даного інструменту відносять:

- зручний у використанні інтерфейс;
- зручність у візуалізації;
- є відкритим для створення нових адаптій та функцій;

– великий набір інструментів для аналізу логів подій.

4.2 Експериментальна перевірка

Згідно з маніфестом логи подій використовують для таких цілей: вилучення, відповідності та удосконалення. Найчастіше лог подій є файлом, котрий містить у собі інформацію про події. Найпоширенішими фрматами є: CSV, MSXML і XES. Нижче на рисунку 4.1 наведено логи подій, які були вилучені з репозиторію.

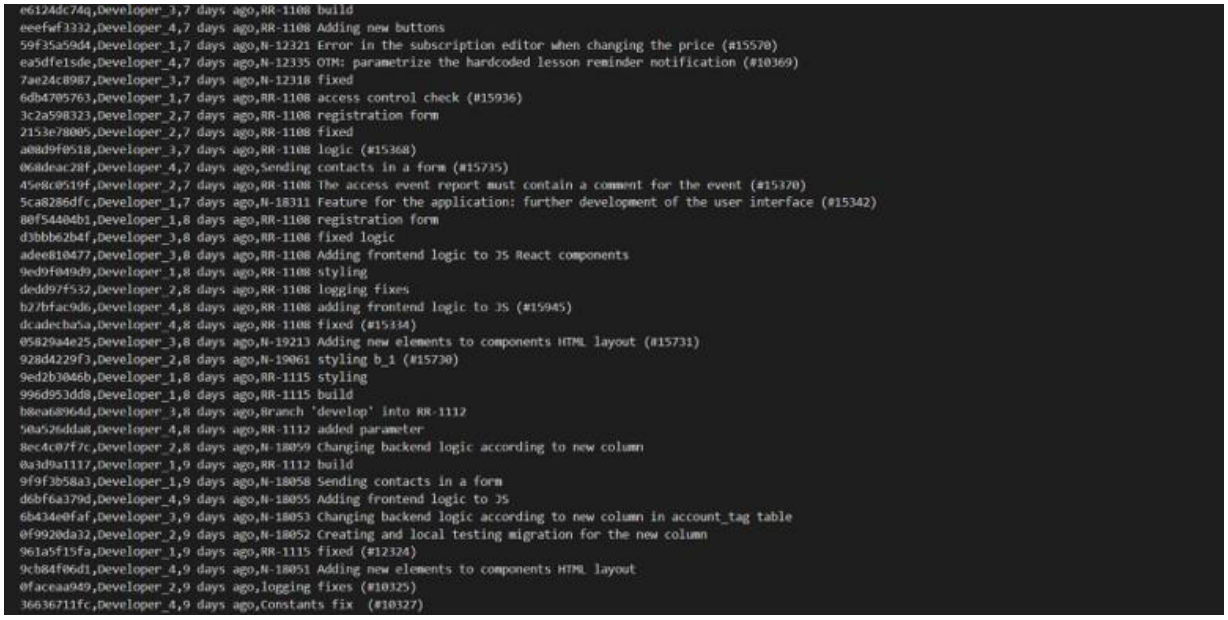

Рисунок 4.1 – Лог розробки

Одним із основних етапів аналізу за допомогою процесних методів є перегляд статистики згідно до даних, які взято із логів подій. Через це виникла

необхідність у виборі інструменту для перевірки бізнес-процесів. Для аналізу та визначення вузьких місць було обрано інструмент ProM 6.13. Він був розроблений для розгортки модулів, де були реалізовані алгоритми процесних методів.

В роботі буде реалізовано нечіткий (Fuzzy) алгоритм. Необхідність його використання виникає, коли необхідно створити спрощену модель, де його основою виступають складні й неструктуровані логи подій. Нечіткий алгоритм є ефективним засобом у візуалізації процесів.

Перед використанням необхідно зробити конвертацію файлу із формату CSV в XES, що наведено на рисунку 4.2.

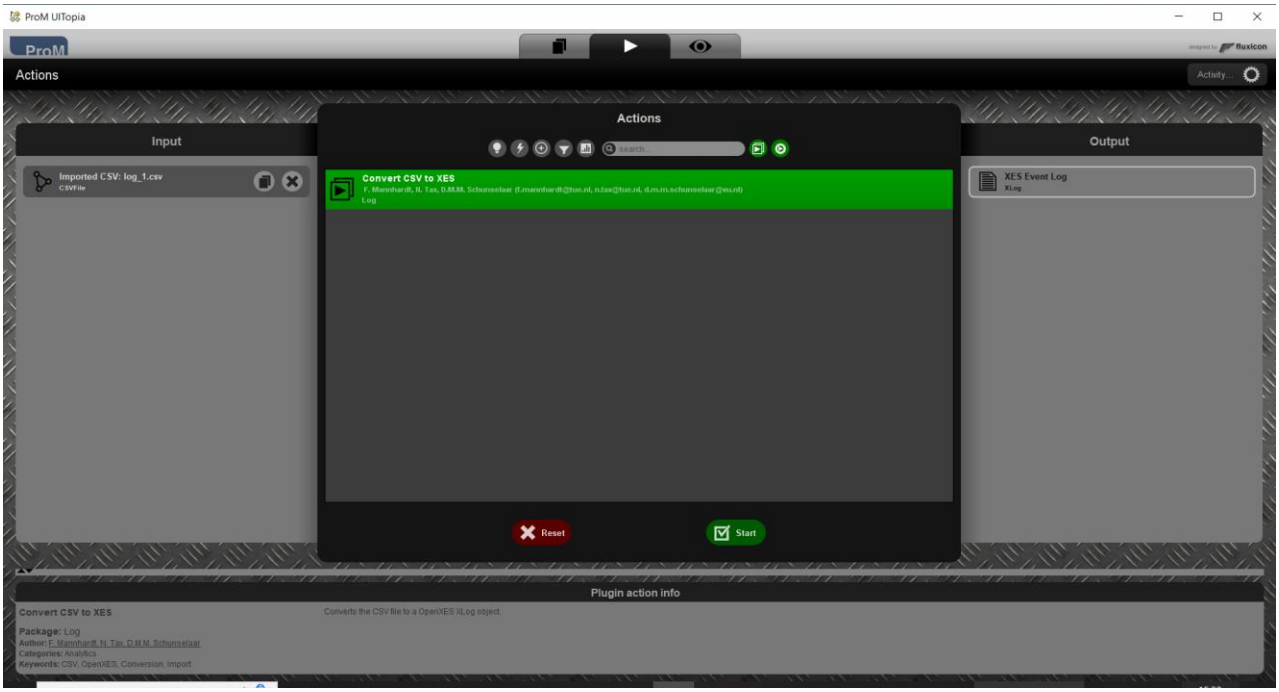

Рисунок 4.2 – Конвертація в потрібний формат

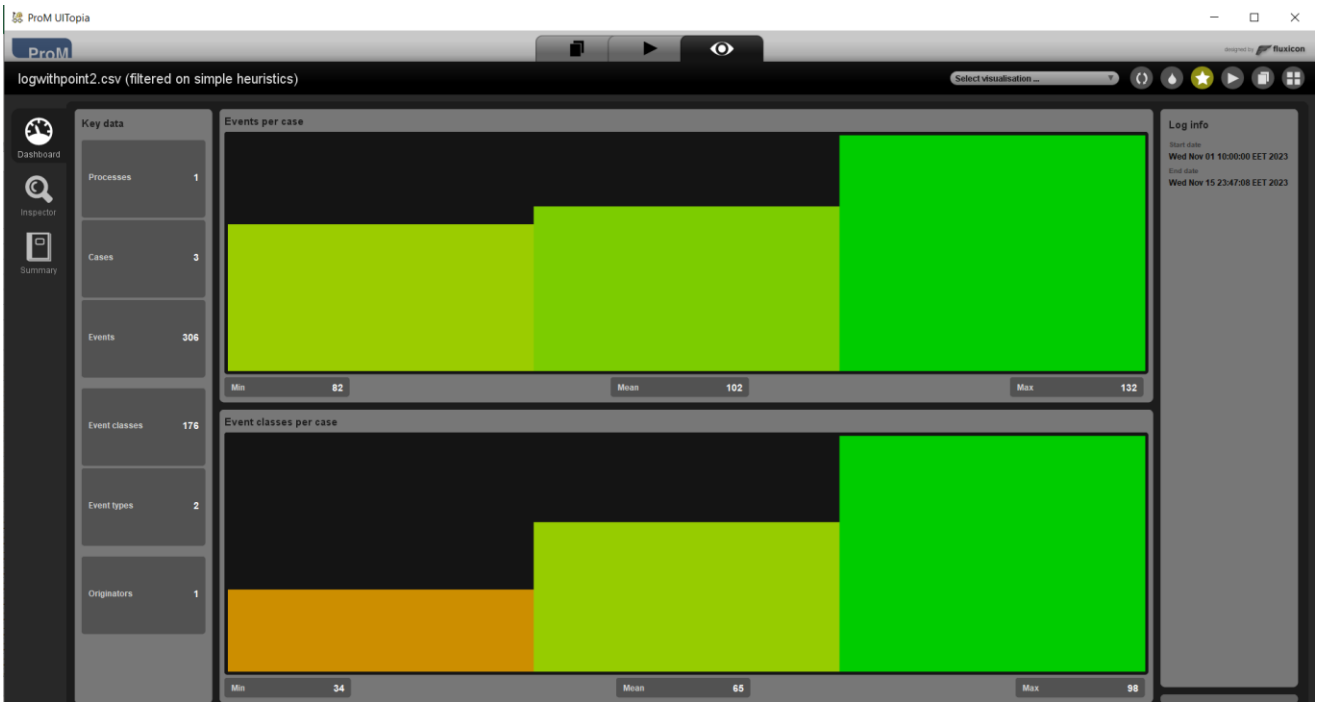

Рисунок 4.3 – Статистичні дані відповідно до логу подій

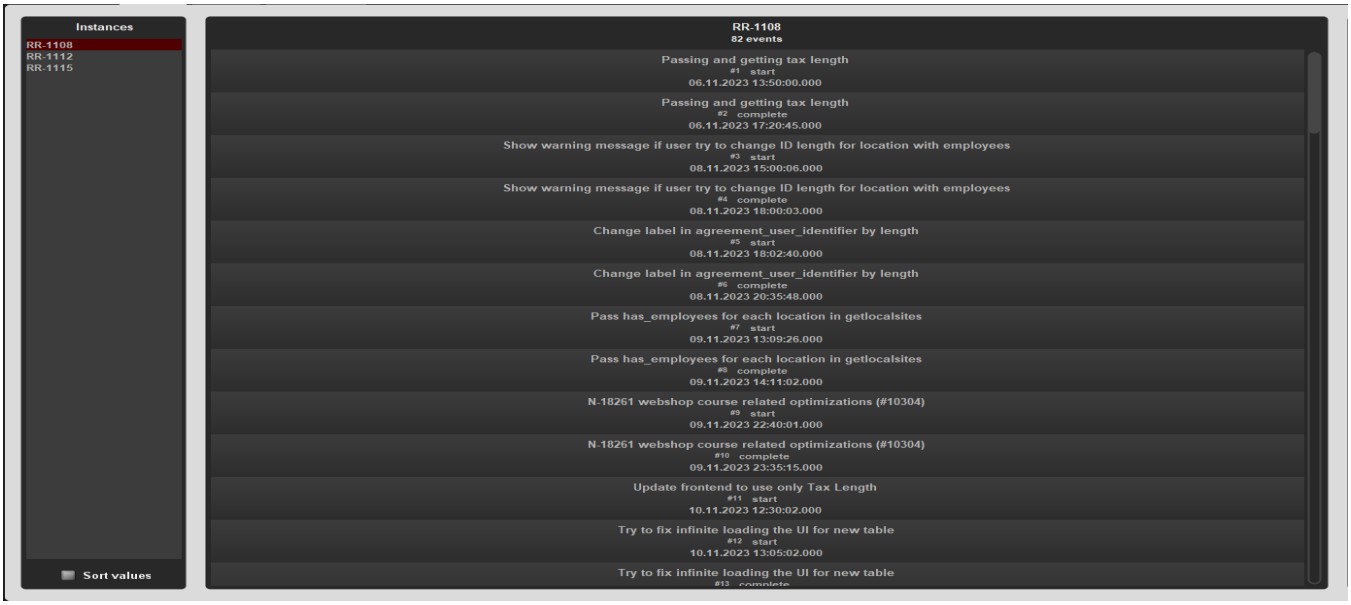

Рисунок 4.4 – Складові логу

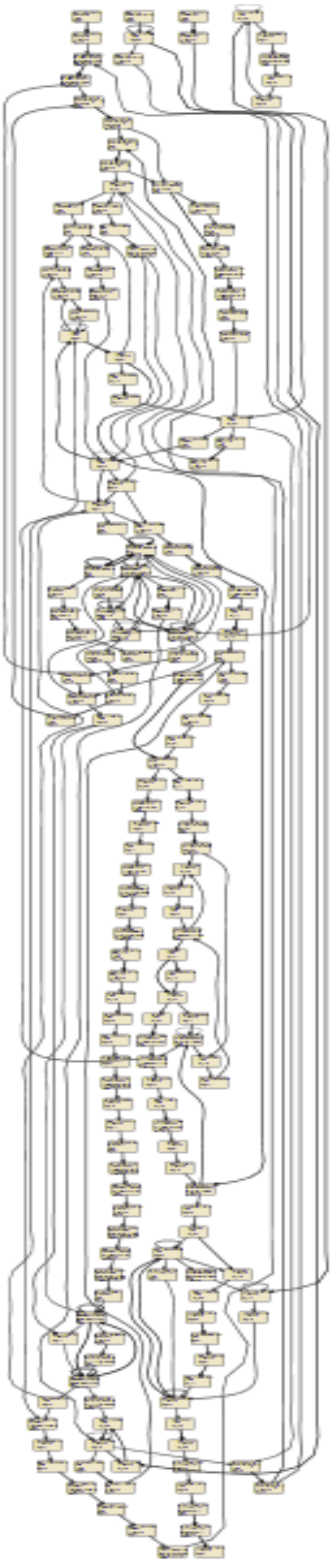

Рисунок 4.5 – Модель Fuzzy в ProM 6.13

Задля аналізу логів подій процесних методів гнучкої розробки використовують побудову різних моделей. У даному випадку застосовано Fuzzy для побудови моделі. Цей алгоритм побудови процесів був створений у 2007 році К. Гюнтером. Алгоритм став відомим завдяки можливості вірішення проблеми великої кількості подій в лозі розробки.

Завдяки ньому було отримано дані стосовно ваг, що показано на рисунку 4.6.

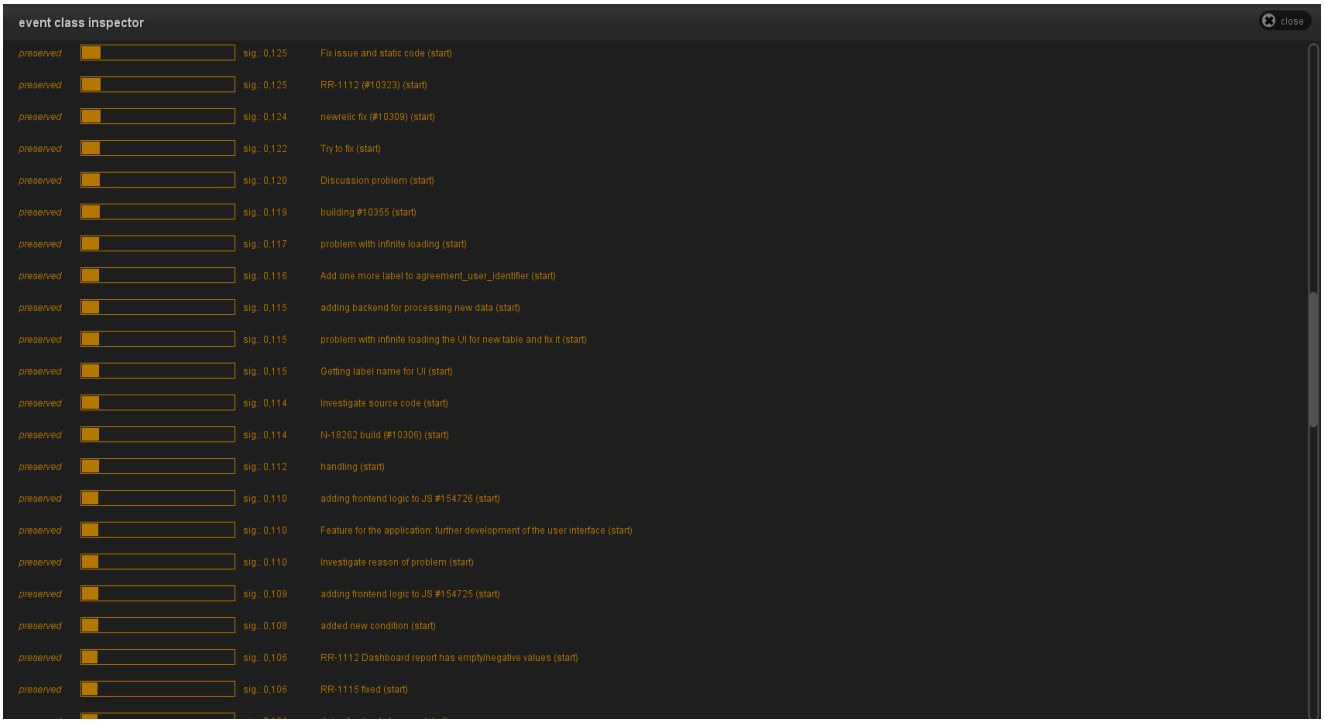

Рисунок 4.6 – Аналіз моделі Fuzzy в ProM 6.13

Таблиця 4.1 – Виявлення вузьких місць

N	Розробник	Час	Активність	Ваги	Як має бути	Як є
1	Developer_1	2023-11-10	Fix issue and static code	0,125	2h	3h30
	Developer_1	2023-11-10	Start split data to different	0,087	1h	1h30
	Developer_1	2023-11-10	Investigate source code Fixed	0,114	2h	3h
	Developer_1	2023-11-10	Change generation link	0,086	2h	2h30
	Developer_3	2023-11-09	Investigate reason of problem	0,11	2h	3h
	Developer_3	2023-11-09	Fix static errors	0,091	1h	1h30
2	Developer_1	2023-11-14	Doing backend changes	0,099	2h	2h30
	Developer_1	2023-11-14	Doing frontend changes	0,104	2h	2h30
3	Developer_3	2023-11-07	RR-1115 build	0,273	2h	3h
	Developer_3	2023-11-07	RR-1115 styling	0,33	2h	3h
4	Developer_2	2023-11-01	RR-1112 build	0,254	2h	2h30
	Developer_2	2023-11-02	Added new condition	0,108	3h	3h30
	Developer_2	2023-11-03	Fixed logic	0,165	1h	2h
	Developer_2	2023-11-06	RR-1112 added type	0,092	1h	1h30
5	Developer_4	2023-11-06	Adding frontend logic to JS	0,126	2h	2h30
	Developer_4	2023-11-06	N33544 registration form	0,092	1h30	2h

Відповідно до таблиці 4.1 видно, що у команди є вузькі місця в процесі розробки програмного забезпечення інформаційних систем. Щоб побачити затримку необхідно порівняти останні дві колонки таблиці. Колонка «як має бути» відповідає даним, що були наведені на рисунках 3.2 і 3.3 у попередньому розділі роботи.

Побудована модель процесів надає змогу отримання із логу подій їх якнайбільш структурованого варіанту.

На етапі ретроспективи було прийняте рішення проаналізувати логи розробки за жовтень для того, щоб виявити можливі причини невкладання в строк.

На рисунку нижче наведено аналіз моделі алгоритму Fuzzy:

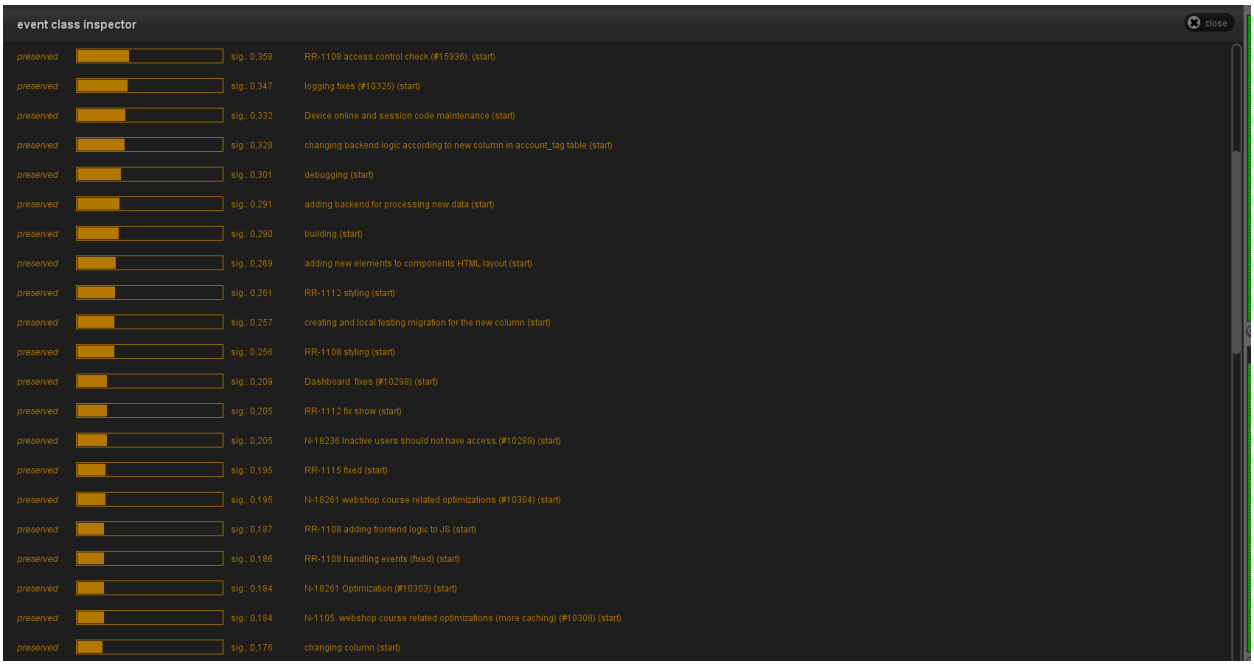

Рисунок 4.7 – Аналіз моделі Fuzzy в ProM 6.13 для 3 проєктів

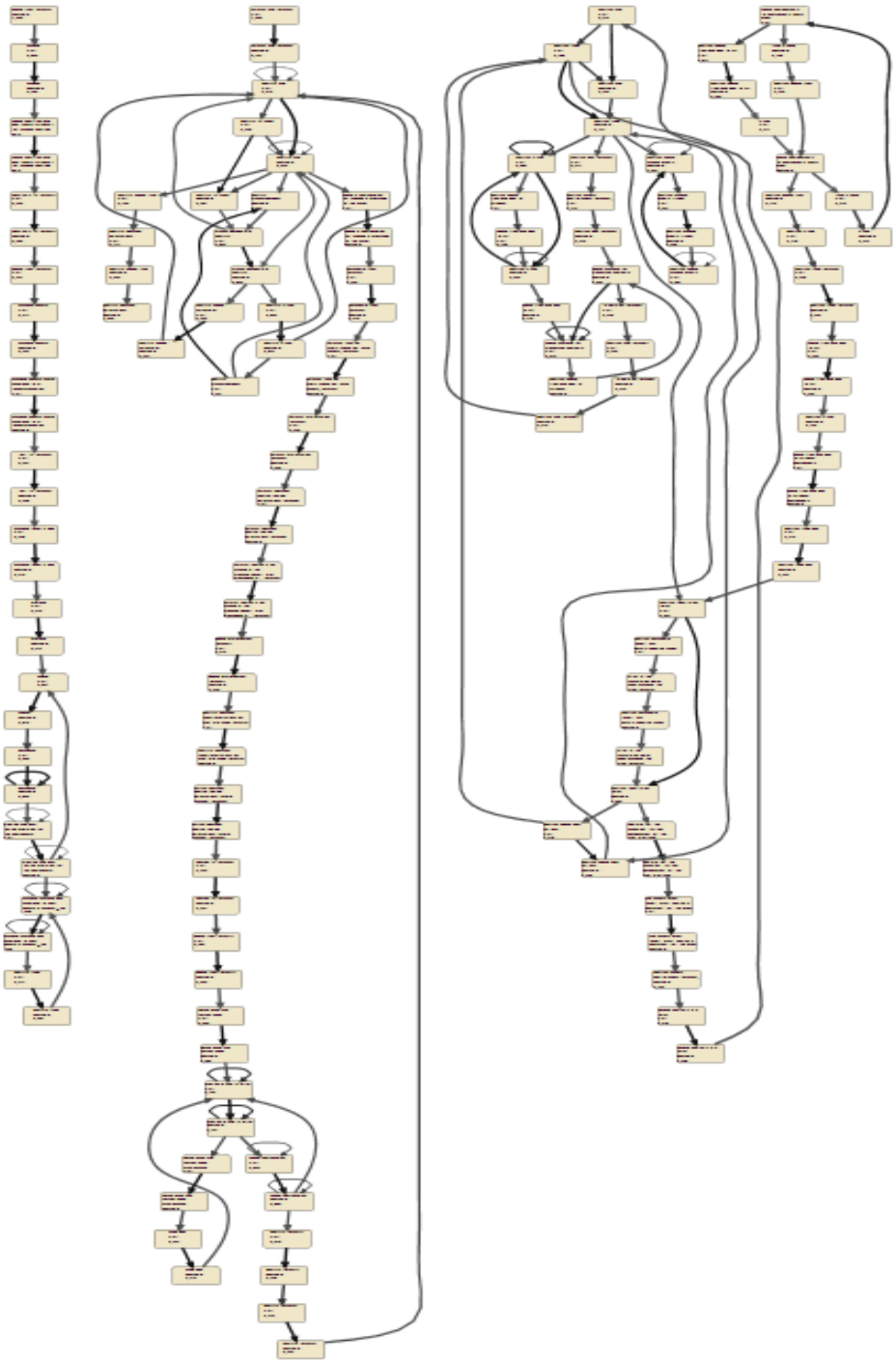

Рис. 4.8 –Побудована модель для 3 проєктів за допомогою алгоритму Fuzzy в ProM 6.13

Таблиця 4.2 – Вузькі місця

N	Розробник	Час	Активність	Ваги	Як має бути	Як є
1	Developer_1	2023-10-18	N-1105 webshop course related optimizations (more caching) #10308	0,184	2h	2h30
	Developer_1	2023-10-18	RR-1108 fixed logic	0,128	1h	1h30
	Developer_1	2023-10-18	Dashboard fixes(#10298)	0,209	1h	2h
	Developer_1	2023-10-16	Changing backend logic according to new column in account_tag	0,328	2h	2h30
2	Developer_4	2023-10-12	RR-1108 registration form	0,378	1h	1h30
	Developer_4	2023-10-12	RR-1108 adding handing events	0,373	1h	2h
	Developer_4	2023-10-12	Adding new frontend logic (HTML/JS/React) for handing new field data	0,135	2h	2h30
3	Developer_2	2023-10-11	Adding a new condition for showing a message to the buyer	0,13	2h	2h30
	Developer_2	2023-10-10	Adding frontend logic to JS	0,137	2h	2h30
4	Developer_3	2023-10-11	RR-1108 logging fixes	0,145	2h	2h30
	Developer_3	2023-10-18	RR-1103 change generation link	0,125	1h30 m	2h
	Developer_3	2023-10-18	Fix the contact form	0,098	1h	1h30

Зважаючи на таблиці 4.1 та 4.2, що відображають вузькі місця логів подій видно, що колонки «як має бути» і «як є» відрізняються незначно. Та все таки виникає необхідність у знаходженні можливих проблем, щоб мінімізувати ризики в майбутньому.

На етапі ретроспективи було наведено такі варіанти причин невкладання в строк спринта:

– занизька кваліфікація усієї команди розробників або її частини;

– низька продуктивність;

– проблема, що стосується комунікації не тільки усередині середовища команди, але із зовнішнім фактором (замовником);

– забагато завдань на одного розробника;

– оптимістична оцінка проєкту на етапі планування (оцінка складності та тривалості).

Було вирішено провести розрахунок часового показника спринтів, що відбулися в листопаді та жовтні.

Нижче наведено розрахунок часового показника за листопад.

Таблиця 4.3 – Розрахований часовий показник за листопад

N	Розробник	Час	Активність	Ваги	Δ t
1	Developer_1	2023-11-10	Fix issue and static code	0,125	1h30
1	Developer_1	2023-11-10	Start split data to different	0,087	30m
1	Developer_1	2023-11-10	Investigate source code Fixed	0,114	1h
1	Developer_1	2023-11-10	Change generation link	0,086	30m
1	Developer_3	2023-11-09	Investigate reason of problem	0,11	1h
1	Developer_3	2023-11-09	Fix static errors	0,091	30m
2	Developer_1	2023-11-14	Doing backend changes	0,099	30m
2	Developer_1	2023-11-14	Doing frontend changes	0,104	30m
3	Developer_3	2023-11-07	RR-1115 build	0,273	1h
3	Developer_3	2023-11-07	RR-1115 styling	0,33	1h
4	Developer_2	2023-11-01	RR-1112 build	0,254	30m
4	Developer_2	2023-11-02	Added new condition	0,108	30m
4	Developer_2	2023-11-03	Fixed logic	0,165	1h
4	Developer_2	2023-11-06	RR-1112 added type	0,092	30m
5	Developer_4	2023-11-06	Adding frontend logic to JS	0,126	30m
5	Developer_4	2023-11-06	N33544 registration form	0,092	30m

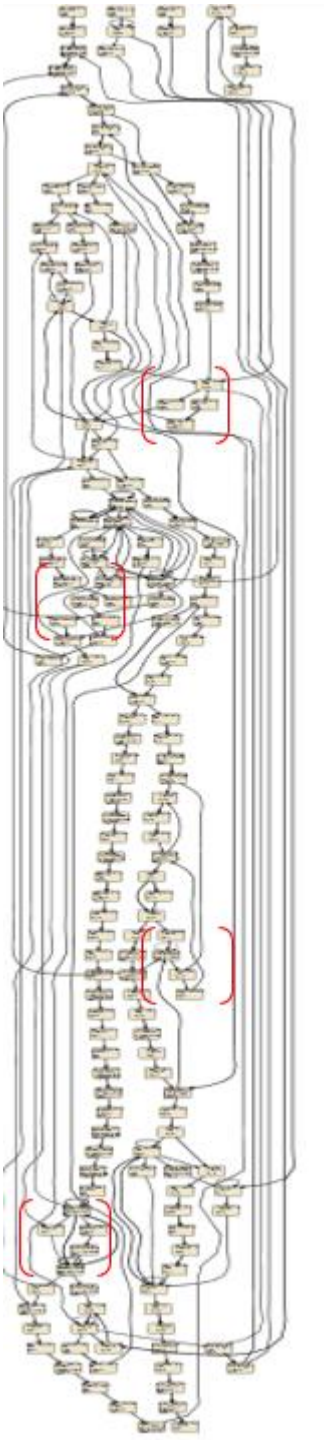

Рисунок 4.9 – Модель Fuzzy в ProM 6.13 з наведеними вузькими місцями

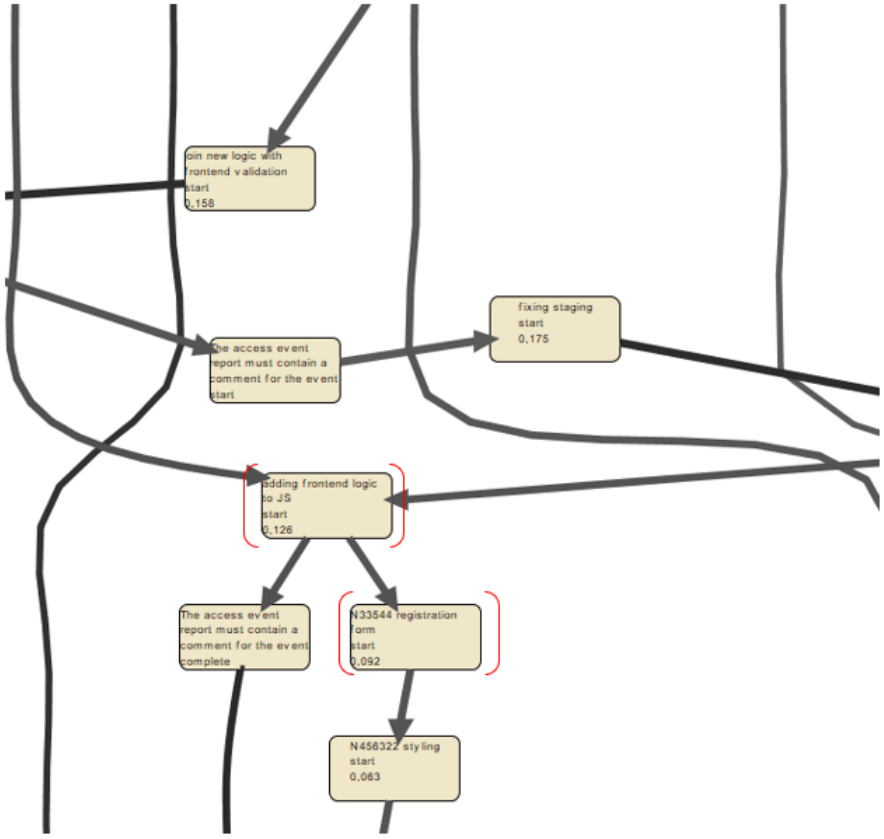

Рисунок 5.1 – Фрагмент моделі Fuzzy в ProM 6.13 з наведеними вузькими місцями

Таблиця 4.4 – Розрахований часовий показник за жовтень

N	Розробник	Час	Активність	Ваги	Δ t
1	Developer_1	2023-10-18	N-1105 webshop course related optimizations (more caching) #10308	0,184	30m
1	Developer_1	2023-10-18	RR-1108 fixed logic	0,128	30m
1	Developer_1	2023-10-18	Dashboard fixes(#10298)	0,209	1h
1	Developer_1	2023-10-16	Changing backend logic according to new column in account_tag	0,328	30m
2	Developer_4	2023-10-12	RR-1108 registration form	0,378	30m
2	Developer_4	2023-10-12	RR-1108 adding handing events	0,373	1h
2	Developer_4	2023-10-12	Adding new frontend logic (HTML/JS/React) for handing new field data	0,135	30m
3	Developer_2	2023-10-11	Adding a new condition for showing a message to the buyer	0,13	30m
3	Developer_2	2023-10-10	Adding frontend logic to JS	0,137	30m
4	Developer_3	2023-10-11	RR-1108 logging fixes	0,145	30m
4	Developer_3	2023-10-18	RR-1103 change generation link	0,125	30m
4	Developer_3	2023-10-18	Fix the contact form	0,098	30m

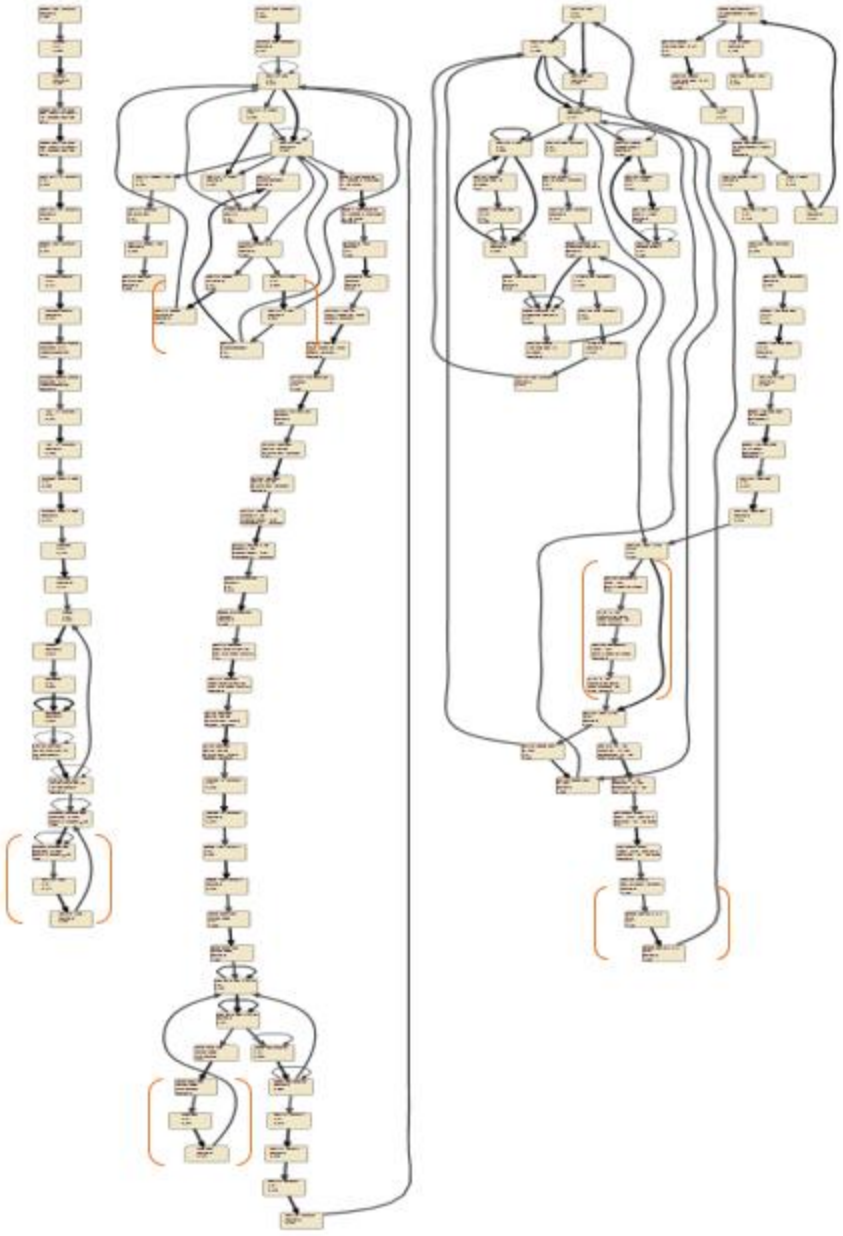

Рисунок 5.2 – Модель Fuzzy для 3 проєктів в ProM 6.13 з наведеними вузькими місцями

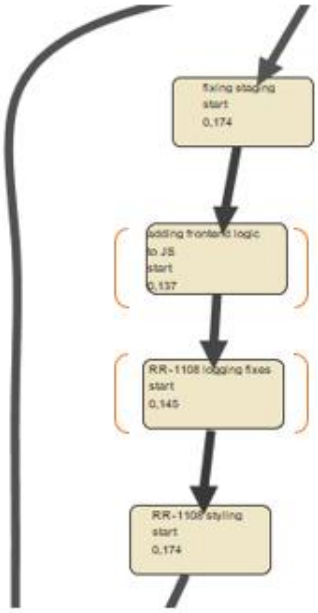

Рисунок 5.3 – Фрагмент моделі Fuzzy в ProM 6.13 з наведеними вузькими місцями

Якщо порівняти таблиці 4.3 і 4.4 бачимо, що деякі проблеми у активностях збігаються. Для Developer_1 у жовтні була поставлена такі активності з якими розробник не справився в строк: N-1105 webshop course related optimizations (more caching) #10308, RR-1108 fixed logic. А вузькі місця за листопад були такі: Start split data to different, Investigate source code Fixed, Change generation link, Doing backend changes, Doing frontend changes. Тож із цього можна зробити висновок, що для даного випадку відповідає варіанту E, бо деякі проблеми були схожу, а деякі – ні.

Порівнявши активності, де були затримки розробника Developer_2 бачимо, що в жовтні виникли такі проблемні місця: Adding a new condition for

showing a message to the buyer, Adding frontend logic to JS. У листопаді виникли затримки з RR-1112 build, Added new condition, Fixed logic, RR-1112 added type. Тож даний випадок теж належить до варіанту E.

Для Developer_3 активність за жовтень складає: RR-1108 logging fixes, RR-1103 change generation link, Fix the contact form. За листопад була така активність: Investigate reason of problem, Fix static errors, RR-1115 build, RR-1115 styling. Тож оскільки активності іноді були схожими, а іноді – різними, це відноситься до варіанту E.

У Developer_4 за жовтень були виявлені вузькі місця: RR-1108 registration form, RR-1108 adding handing events, Adding new frontend logic (HTML/JS/React) for handing new field data. За листопад були такі вузькі місця: Adding frontend logic to JS, N33544 registration form. Тож тут виконується варіант E.

У даному дослідженні було використано метод побудови темпоральної моделі процесу розробки програмного забезпечення. Як результат бачимо, що проблемні місця інколи були подібними, але частіше у розробників виникали проблеми різнопланового характеру. Затримки у проблемних місцях трималися від 30 хвилин до 1 години. Це показує, що команда розробки є професіоналами у своїй справі. Та причина виникає не через компетентність у окремому завданні одного розробника, а через великої завантаженості фахівців.

На етапі ретроспективи були запропоновані такі варіанти: додати в команду ще одного розробника. Це трохи розвантажить команду. Та допоможу в уникненні подальших можливих ризиків.

ВИСНОВКИ

Отже, на даному етапі дослідження можна зробити висновок, що усі методи, підходи та технології, котрі були відмічені в даній роботі спрямовані на тільки на створення якісного кінцевого програмного продукту, а й ефективної співпраці команди розробників.

Стрімкий розвиток інформаційних систем і технологій, який різко зріс останнім часом розуміють як збільшення інформаційних потоків через процеси глобалізації.

На даний момент процес управління у компаніях має на увазі використання інформаційних систем і технологій, адже вони надають змогу у підвищенні ефективності. Тож для продуктивної діяльності команд необхідне постійне удосконалення процесів, адже це сприятиме підвищенню конкурентоздатності на ринку компаній.

Оскільки мінімізація часу та трудовитрат допомагає у підвищенні результативності управлінських та продуктивних процесів, компаніям потрібно застосовувати у своїй діяльності нові підходи для підвищення рентабельності, а це у свою чергу сприяє зниженню затрат та збільшення прибутку.

При написанні даної роботи було проведено дослідження процесних методів гнучкої розробки програмного забезпечення інформаційних систем. Було проаналізовано процес розробки програмного забезпечення інформаційних систем, проаналізовано Agile методи та підходи інформаційних систем. Досліджено процесні методи. Використано темпоральні правила при аналізі розробки логу подій. Проведена експериментальна перевірка дослідження.

Результат цього дослідження направлений на привернення уваги фахівців даної галузі до його продовження та удосконалення. Отриманий результат допоможе команді розробників на етапі ретроспективи, що у своїй роботі використовують методологію Scrum. Окрім цього варто розглянути в майбутньому варіанти застосування цього дослідження у інших областях, де є потреба у аналізі процесів.

За результатами досліджень магістерської атестаційної роботи підготовлено тези у вигляді доповіді на тему «Аналіз гнучких методологій розробки програмного забезпечення», що представлено у матеріалах IX Міжнародної молодіжної науково-практичної інтернет-конференції «Наука і молодь в XXI сторіччі» 2023 р.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки щодо розробки та оформлення кваліфікаційної роботи (для студентів усіх форм навчання другого (магістерського) рівня вищої освіти спеціальності 122 Комп'ютерні науки освітньо-професійної програми «Інформаційні управляючі системи та технології») / Упоряд.: Петров К.Е., Левикін В.М., Чалий С.Ф., Євланов М.В., Саєнко В.І., Міхнов Д.К., Міхнова А.В., Чала О.В. – Харків: ХНУРЕ, 2021. – 30 с.

2. ДСТУ 3008-2015. Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлювання. – Чинний від 04.03.2016. – Київ: ДП «УкрНДНЦ», 2016. – 31 с.

3. ДСТУ 8302-2015. Інформація та документація. Бібліографічні посилання. Загальні положення та правила складання. – Чинний від 04.03.2016. – Київ: ДП «УкрНДНЦ», 2016. – 20 с.

4. Google Trends. URL: https://trends.google.ua.

5. PMI Ukraine. URL: https://pmiukraine.org.

6. Азаров М.Я., Ярошенко Ф.О., Бушуєв С.Д. Інноваційні механізми управління програм розвитку / М.Я. Азаров, Ф.О. Ярошенко, С.Д. Бушуєв – К.: Самміт книга, 2011. – 564 с.

7. Андерсон Дж. Kanban. – Х.: Фабула, 2021. -288 с.

8. Левикін В.М., Чала О.В. Виділення реляційних залежностей бізнес-процесу на основі аналізу його логу. Національний авіаційний університет. 2016. №4. С. 405-409.

9. Левикін В.М., Чала О.В. Метод підтримки управлінських рішень в умовах невизначеності на основі темпоральних знань. Біоніка

інтелекту. 2018. № 91. С. 54-59.

10. Левикін В.М.,Чала О.В. Модель бази знань інформаційної системи процесного управління. Вісник Національного технічного університету «ХПІ». Системний аналіз, управління та інформаційні технології. 2017. №28 (1250). С. 74-78.

11. Левикін В.М., Чала О.В. Підхід до визначення аномальної поведінки процесів в системах процесного управління на основі аналізу логів. Вісник Національного технічного університету «ХПІ». 2017. № 55 (1276).

12. Мінцберг Г. Зліт та падіння стратегічного планування / Г. Мінцберг. – К.: Вид-во Олексія Капусти. – Київ, 2008. – 389 с.

13. Окерман С. Опанування професійного Scrum. – Київ : RANOK, 2023. – 224 с.

14. Павлова С.І. Проєктно-орієнтовані організації як розвиток методів управління підприємством // Вісник ЖДТУ. Серія: Економічні науки. 2016. №4 (78). – С. 170 - 177.

15. Петренко Н.О., Кустріч Л.О., Гоменюк М.О. Управління проєктами / Н.О. Петренко, Л.О. Кустріч, М.О. Гоменюк. – Київ. 2015. – 257с.

16. Піхлер Р. Agile продукт-менеджмент за допомогою Scrum. Київ: Фабула, 2022. – 128 с.

17. Сазерленд Дж. Scrum. Навчись робити вдвічі більше за менший час. – К.: КСД, 2022. – 280 с.

18. Чала О.В., Буцукіна І.Б. Ідентифікація процесів підприємства при впровадженні процесно-орієнтованої системи управління якістю. Вісник економіки транспорту і промисловості. 2009. С. 255-258.

19. Чала О.В. Інформаційний простір підприємства як об’єкт

управління. 2010.

20. Чала О.В. Модель узагальненого представлення темпоральних знань для задач підтримки управлінських рішень. Вісник Національного технічного університету «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. 2020. №1(3) С. 14-18.

21. Чала О.В. Побудова темпоральних правил для представлення знань в інформаційно-управляючих системах. Національний технічний університет «Харківський політехнічний інститут». 2018.

22. Чала О.В. Поліпшення процесів системи управління якістю на основі коригувальних і попереджувальних дій. Вісник економіки транспорту і промисловості: Зб. наук. праць. – Харків: УкрДАЗТ, 2006. – Вип. 15-16. С 118-121.

23. Чалий С.Ф., Прібильнова І.Б. Побудова ситуаційного представлення знань на основі аналізу логів. Вісник Національного технічного університету «ХПІ»: Системний аналіз, управління та інформаційні технології. 2017. № 28. С. 70-73.

24. Якобсон А., Буч Г., Рамбо Дж. Уніфікований процес розробки ПЗ / А. Якобсон., Г. Буч., Дж. Рамбо. – Київ, 2014. – 300 с.