Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Метод інтелектуального аналізу процесів для розробки проектів
(тема)

Виконав:
студент 2 курсу, групи СШІм-20-1
Кізілов М.С.
(прізвище, ініціали)

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

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

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник доц. Золотухін О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Кізілову Михайлу Станіславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Метод інтелектуального аналізу процесів для розробки проектів

затверджена наказом університету від 8 листопада 20 21 р. № 1695Ст

2. Термін подання студентом роботи до екзаменаційної комісії _____ _____ 20 21 р.

3. Вихідні дані до роботи методології управління проектами, , гнучкі методології, інформаційна технологія, система управління проектами, спрінт, Agile, Kanban, Scrum, стандарти ISO

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної області і постановка задачі
2) Принципи та методи управління розробкою програмного забезпечення
3) Оптимізація вибору і адаптація методів управління процесами розробки програмного забезпечення
4) Практичне використання результатів дослідження

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	к.т.н., доцент Золотухін О.В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Отримання завдання на кваліфікаційну роботу	01.09.2021	виконано
2.	Аналіз завдання та пошук літератури за темою	02.09.2021–08.09.2021	виконано
3.	Опрацювання літератури та аналіз об'єкту	09.09.2021–15.09.2021	виконано
4.	Вибір програмних засобів для розробки системи	16.09.2021–18.09.2021	виконано
5.	Розробка програмного засобу	19.09.2021–05.11.2021	виконано
6.	Аналіз отриманих результатів	06.11.2021–16.11.2021	виконано
7.	Оформлювання пояснювальної записки	17.11.2021–24.11.2021	виконано
8.	Проходження нормоконтролю	25.11.2021–28.11.2021	виконано
9.	Оформлення презентаційних матеріалів	29.11.2021–01.12.2021	виконано
10.	Попередній захист	02.12.2021	виконано
11.	Представлення кваліфікаційної роботи		

Дата видачі завдання 1 вересня 2021 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 116 с., 34 рис., 1 табл., 1 дод., 27 джерел.

БЕКЛОГ, ГНУЧКІ МЕТОДОЛОГІЇ, ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, СИСТЕМА УПРАВЛІННЯ ПРОЕКТАМИ, СПРІНТ, AGILE, KANBAN, SCRUM, TOUTRACK.

Об'єкт дослідження – гнучкі методи управління проектами з розробки програмного забезпечення.

Предмет дослідження – Agile-методологія управління процесом розробки ПЗ.

Мета кваліфікаційної роботи – за результатами аналізу предметної області, визначити особливості гнучких методологій управління проектами, розробити шляхи покращення вибору Agile-методології управління IT-проектами з розробки ПЗ за рахунок використання методів Process mining.

В кваліфікаційній роботі виконано аналіз методів управління проектами з вкористанням гнучкої методики управління Scrum. Проаналізовано методи управління проектами в Agile та сучасні технологічні засоби управління для автоматизації IT- проектів; розроблено алгоритм виборі методології швидкої розробки відповідно до вимог; вибрано підходяще програмне забезпечення для реалізації проекту; визначено особливості проектної діяльності в IT-компаніях, запропоновано метод виявлення вузьких місць в організації проекту з використанням реросpектив.

Область застосування – менеджмент управління процесом розробки програмного забезпечення в IT-компаніях.

РЕФЕРАТ

Пояснительная записка: 116 с., 34 рис., 1 табл., 1 прил., 27 источников.

БЕКЛОГ, ГИБКИЕ МЕТОДОЛОГИИ, ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ, СИСТЕМА УПРАВЛЕНИЯ ПРОЕКТАМИ, СПРИНТ, AGILE, KANBAN, SCRUM, TOUTRACK.

Объект исследования – гибкие методы управления проектами по разработке программного обеспечения.

Предмет исследования – Agile-методология управления процессом разработки ПО.

Цель квалификационной работы – по результатам анализа предметной области определить особенности гибких методологии управления проектами, разработать пути улучшения выбора Agile-методологии управления ИТ-проектами по разработке ПО за счет использования методов Process mining.

В квалификационной работе выполнен анализ методов управления проектами с использованием гибкой методики управления Scrum. Проанализированы методы управления проектами в Agile и современные технологические средства управления для автоматизации IT-проектов; разработан алгоритм выбора методологии быстрой разработки в соответствии с требованиями; выбрано подходящее программное обеспечение для реализации проекта; определены особенности проектной деятельности в ИТ-компаниях, предложен метод выявления узких мест в организации проекта с использованием переспектив.

Область применения – менеджмент управления процессом разработки программного обеспечения в IT-компаниях.

ABSTRACT

Explanatory note: 116 p., 34 fig., 1 tabl., 2 ann., 27 sources.

AGILE, BECKLING, FLEX METHODOLOGY, INFORMATIONAL TECHNOLOGY, KANBAN, PROJECT MANAGEMENT SYSTEM, SCRUM, SPRINT, TOUTRACK.

Object of research – flexible methods of project management software development.

Subject of the study – Agile-methodology for managing the software development process.

The purpose of certification work – based on the results of the domain analysis, to identify the features of flexible project management methodologies, to develop ways to improve the choice of Agile-methodology for managing IT software development through the use of Process mining methods.

In this work the analysis of methods project management with the use of the flexible method management of Scrum is carried out. The methods of project management in Agile and modern technological tools for automation IT projects are analyzed; the algorithm of the choice the rapid development methodology is developed in accordance with requirements; suitable software for project implementation was selected; the peculiarities of the project activity at IT companies are determined, the method of identifying bottlenecks in the organization of the project with the use of retrospective methods is proposed.

Scope –management of the software development process in IT companies.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІАД – інтелектуальний аналіз даних;

ІАП – інтелектуальний аналіз процесів;

ЖЦ – життєвий цикл;

ПЗ – програмне забезпечення;

ASD – Adaptive Software Development – адаптивна розробка програмного забезпечення;

RUP – Rational Unified Process – раціональний уніфікований процес;

UML – Unified Modeling Language – уніфікована мова моделювання;

ВСТУП

Розробка програмного забезпечення – швидко зростаюча галузь, вплив якої на глобальну економіку істотно перевершує її долю у світовому виробництві. Використання програмного забезпечення значно впливає на продуктивність праці в усіх галузях. На теперішній час інформаційні технології є основним каталізатором управлінських інновацій, що забезпечують економічне зростання.

Програмне забезпечення (ПЗ) є ядром технологій, що формують інформаційне суспільство. Інформаційне суспільство визначається як суспільство, в якому процес комп'ютеризації надає людству доступ до надійних джерел інформації, позбавляє їх рутинної роботи, забезпечує високий рівень автоматизації виробництва. При цьому продукт стає здебільш інформаційно містким [1].

Аналізуючи економічний і соціальний ефект від застосування інформаційних технологій, їх вплив на життя суспільства і визначаючи перспективи розвитку технологій, фахівці акцентують увагу на проблемі розробки ПЗ як на один зі стримуючих чинників: «Програмне забезпечення саме по собі слід розглядати як одне з найбільших і вкрай складних досягнень людського розуму. При його створенні кожну дію, що вимагається від комп'ютера, слід перетворити в набір специфічних машинних команд. Не дивно, що до програмування залучається все більша частина сукупного інтелектуального потенціалу планети. І зараз, як і 60 років тому, комп'ютерне програмування залишається процесом виключно трудомістким, складним, відповідальним і навіть тяжким. Розмови про підвищення рівня програмування і гарантії якості кінцевого продукту ведуться в серед фахівців вже більше 40 років, але вони навряд чи наблизили нас до вирішення проблеми» [2].

Проектний характер розробки ПЗ та високий рівень ризиків обумовили розвиток спеціалізованих проектних методологій, покликаних забезпечити якість ПЗ, підвищити продуктивність праці розробників і понизити проектні риски за рахунок повторюваності процесів, використання кількісних методів

оцінок і наявності процедур контролю і оптимізації. Це робить необхідним використання спеціалізованих методів управління розробкою, що і обумовлює актуальність вибраної теми кваліфікаційної роботи.

Технологія створення ПЗ залишає значний ступінь свободи у виборі різних варіантів організації процесу розробки і підходів до управління. Проектний характер розробки ПЗ і високий рівень ризиків зумовили розвиток спеціалізованих проектних методологій, а також використання методів управління ризиками, покликаними забезпечити якість ПЗ за рахунок повторюваності процесів, використання кількісних методів оцінок, наявності процедур контролю і оптимізації тощо.

Існує проблема теоретичного узагальнення і використання досвіду, накопиченого в індустрії розробки ПЗ. Відсутність структурованого підходу до управління і використання неефективних методологій характерна для багатьох компаній розробників ПЗ. Ця кваліфікаційна робота орієнтована на розв’язання позначених проблем.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

Дослідження в сфері теорії управління розробкою ПЗ проводилися за двома основними напрямами. З одного боку, розроблялися методології, що ґрунтуються на концепціях теорії управління та є її спеціалізацією, що враховує особливості процесів управління розробкою ПЗ. У цьому напрямі веде роботу інститут Картеги Мелона (Carnegie Mellon University Software Engineering Institute, CMU SEI), а також провідні компании з розробки ПЗ: IBM, HP, Microsoft, SAP, Oracle. З іншого боку, відбувається безперервне теоретичне узагальнення практичного досвіду реалізації IT-проектів.

Еволюція підходів до управління розробкою ПЗ обумовлена розвитком технологій розробки і накопиченням досвіду. Це призвело до формування теорії управління розробкою ПЗ в її сучасному вигляді. З'явилися комерційні компоненти повторного використання, операційні системи, бази даних, мови високого рівня, автоматизовані засоби аналізу, проектування і розробки. Серед значимих подій в історії розвитку теорії управління розробкою ПЗ можна виділити наступні події [3]:

1968 р. Уїнстон Ройс описав дві основні моделі життєвого циклу розробки ПЗ: каскадну і ітеративну моделі [3].

1975 р. Фредерік Брукса у своїй роботі описав основні проблеми проектів розробки ПЗ і запропонував їх рішення. Він звернув увагу на істотні відмінності в продуктивності окремих розробників і запропонував ефективну модель команди проекту.

1977 Алан Албрехт (IBM) запропонував метрику Function Point (Функціональні точки) для оцінки об'єму ПЗ. Метрика FP дозволила кількісно оцінювати об'єм роботи на ранніх стадіях проекту розробки ПЗ.

1979 р. Флетчер Баклей (Fletcher Buckley) з IEEE Computer Society ініціював розробку стандарту забезпечення якості ПЗ – IEEE 730.

1979 р. Крістофер Александер розробив теорію «патернів».

1993 р. Кен Швабер і Майкл Бідл запропонували методологію швидкої розробки Scrum і підхід «Адаптивний процес швидкої розробки» (Agile Software Development).

1994 р. Уотс Хэмфри запропонував «Персональний процес розробки» (PSP, Personal Software process).

1994 р. Джим Рабо, Граді Буч та Айвар Якобсон (three amigos & Rutional Software Corporation) об'єднали свої об'єктні нотації і створили UML (Unified Modelling language, уніфікована мова розробки).

1995 р. Філіп Крачтен (Rational Software Corporation) розробив «Раціональний уніфікований процес» (RUP, Rational Unified Process).

1996 м. Кент Бек запропонував методологію «екстремального програмування» (XP, eXtreme Programming).

2001 р. Група прибічників методології Agile Software Development Кент Бек, Алістер Коберн, Джим Хайсміт та ін. опублікували маніфест швидкої адаптивної розробки ПЗ «Agile Manifesto».

Ф. Брукс сформулював проблему оцінки продуктивності праці у сфері розробки програмного забезпечення і визначив основні напрями підвищення продуктивності праці розробників.

Б. Боем розробив сукупність методів прогнозування, оцінки трудомісткості і тривалості проектів розробки програмного забезпечення (метрики), що дістали назву COCOMO (Constructive Cost Model).

Р. Томсетт здійснив аналіз основних переговорних моделей, що використовувались під час визначення вимог, термінів і необхідних ресурсів для програмних проектів.

Різноманітність створених методик, стандартів і методологій зумовило необхідність дослідження підходів адаптивного управління процесом розробки програмних проектів.

1.2 Основні проблеми вибору методів, моделей і стандартів управління розробкою програмних проектів

Застосування сучасної теорії управління для управління програмними проектами і компаніями-розробниками припускає рішення задачі вибору і адаптації методів, моделей і стандартів.

З проблемою створення ефективної системи управління [4] тісно пов'язані такі питання, як:

– створення системи навчання менеджерів і розробників основним поняттям теорії управління розробкою ПЗ;

– впровадження в практику роботи технічних засобів, необхідних для управління розробкою ПЗ;

– використання узгодженого глосарію предметної області, розуміння термінів в області управління розробкою ПЗ, що забезпечує єдність;

– розуміння вимог зовнішнього середовища до організації процесів розробки (наприклад, вимоги до сертифікації за CMM/CMMI з боку клієнтів);

– створення системи управління використанням інструментальних засобів, включаючи інтеграцію з системою фінансового управління компанією. Необхідно планувати витрати на впровадження і використання методів, моделей і стандартів і оцінювати їх ефективність.

Вибір адекватних проектних методологій, моделей ЖЦ ПЗ, метрик проектів, інших інструментальних засобів управління ПЗ є складним завданням для компаній-розробників ПЗ і проектних команд. Складність завдання обумовлена великою кількістю інструментальних засобів, складними взаємозв'язками між ними, а також безперервним розвитком теорії управління процесом розробки ПЗ.

Підходи до вибору методологій, моделей і стандартів залежать від передбачуваної мети використання [5]. Вони можуть застосовуватися таким чином:

– як еталон – так звана «ефективна методологія – срібна куля»;

– для обґрунтування існуючої практики управління проектами.

– як доказ правильності прийнятих рішень;

– як керівництво по досягненню цілей «методологія»;

– як набір вимог, що пред'являються зовнішнім середовищем: клієнтами, партнерами, власниками, державними організаціями, професійними об'єднаннями – «стандарт».

– для пояснення існуючої практики. Використовується термінологія, концепції, ідеї – «модель».

– для забезпечення «ефективного» діалогу між менеджментом, учасниками проектних команд, клієнтами та субпідрядниками – «глосарій».

– для зміни фокусу – центру докладання управлінських зусиль, інвестицій в методології.

Залежно від цілей використання інтерес представляють різні смислові частини методологій, моделей і стандартів. Смисловими частинами є:

– термінологія і визначення;

– ідеї і концепції;

– довідники і класифікації;

– моделі процесів;

– склад і структура документації;

– алгоритми оцінки об'ємів, складності і інших параметрів проекту;

– моделі команд;

– історії успіху, приклади.

Обґрунтування рекомендацій методології за допомогою визнаних концепцій управління робить методології корисними для підтвердження правильності існуючих практик, співпадаючих з рекомендованими. Наявність розгорнутих рекомендацій, що включають опис процесів, моделей команд, вимог до робочих продуктів необхідно для використання методологій у якості

керівництва. Глосарій і класифікації використовуються для будування ефективного діалогу в IT-проектах.

Мета використання методології визначає ефективність її вибору у кожному окремому випадку. Наприклад, CMMI може бути обрана для цілей побудови діалогу, використання загальної термінології і в якості «ідеалу» організації процесів в компанії, але не підходити як практичне «керівництво до дії» для управління IT-проектом. Вона носить занадто абстрактний характер.

Передумовою успішного рішення задачі вибору методології є наявність необхідної і достатньої інформації про неї саму і практику її використання, і додаткових рекомендацій по застосуванню.

Постійний розвиток теорії управління і вдосконалення технологій розробки призводять до вироблення нових принципів, на основі яких створюються моделі і стандарти. Модель бізнесу компанії, її стратегія, культура компанії, особливості виконуваних сьогодні проектів (технології, риски, доступність фінансових ресурсів) визначають вибір інструментальних засобів.

Складність вибору обумовлена [6]:

– великим об'ємом методологій, складністю їх структури;

– безперервним розвитком теорії управління розробкою ПЗ, створенням нових методологій, стандартів, моделей, техніки;

– відсутністю однозначного критерію вибору методологій;

– об'єктивною необхідністю використання різних методологій для різних проектів в компанії.

Для розв’язання задачі ефективного використання існуючих методів, стандартів, моделей необхідно зробити їх класифікацію, провести порівняльний аналіз, визначити, які завдання вони вирішують та яка сфера їх застосування.

Досить часто використовується слабко-структурований підхід до вибору методологій, при якому змішуються методології, які мають різне призначення.

В теорії управління розробкою ПЗ моделі процесів стають важливими об'єктами стандартизації. Особливості процесного підходу до управління призводять до підвищення значущості формалізації процесів, вибраної моделі

процесів. Аналіз моделей процесів дозволяє виявити різницю в підходах до організації роботи IT компанії.

Для розв'язання задачі ефективного вибору методів, моделей і стандартів доцільно виконати аналіз існуючих інструментальних засобів і виявити їх схожі і відмітні риси (рисунок 1.1). Розгляд схожості і відмінностей передбачається виконати в наступних напрямах: історія створення, призначення структура, зміст складових частин.

Першим кроком в рішенні задачі вибирання інструментальних засобів повинна стати їх класифікація.

Рисунок 1.1 – Моделі розробки ПЗ

Необхідною умовою ефективного рішення задачі вибору є усвідомлення можливості і необхідності завдання вибору. На практиці, IT-компанії і проектні команди далеко не завжди уявляють собі всі існуючі можливості вибору інструментальних засобів, розглядають їх обмежений набір, підходять до вирішення проблеми вибору без урахування специфіки компанії і проекту.

Менеджмент і учасники проектних команд повинні сформувати єдине розуміння відносно цілей і методів управління IT-проектами. Таке розуміння

може сформуватися в результаті безперервного сумісного навчання. В роботі [4] пропонується надати можливість розробникам вибирати проектну методологію самостійно. Вирішення проблеми вибору вимагає двох типів компетенцій:

– по-перше, компетенцій в сфері методологій;

– по-друге, компетенцій в області вибору підходів.

Для організації ефективного діалогу в ході вибору методологій, учасники повинні використати єдиний підхід і погоджені критерії. Таким чином, завдання ефективного вибору розпадається на задачу створення множини вибору та задачу створення методології вибору.

За допомогою класифікації визначимо структуру для подальшого аналізу інструментальних засобів з метою генерації рекомендації з їх вибору.

1.3 Класифікація інструментальних засобів управління розробкою ПЗ

Для створення класифікації методів, моделей і стандартів визначені різні ознаки угрупування і виділені групи за цими ознаками [7]. Розглянемо їх більш детально

За характером обґрунтування рекомендацій поділяються на концептуальні і емпіричні. Концептуальні моделі отримані раціонально-логічним методом, а емпіричні – чуттєво-досвідченим. В основі концептуальних інструментальних засобів лежать концепції теорії менеджменту, такі як процесне управління та реінжиніринг бізнес-процесів, управління проектами, управління якістю. Прикладом концептуальної моделі є модель зрілості технологічних процесів SEI (Capability Maturity Model, CMM). Концептуальною також є методологія PRINCE і раціональний уніфікований процес (Rational Unified Process, RUP).

Емпіричні методології розроблені на основі теоретичного узагальнення успішних практик IT-проектів. Прикладами емпіричних моделей є SCRUM, XP, Crystal.

В залежності від призначення виділяють моделі зрілості та процесні моделі, проектні методології та індивідуальні і групові практики. Модель

зрілості CMMI, модель оцінки процесів SPICE і стандарт ISO 9000 використовуються для управління IT-компанією. Проектні методології MSF, SCRUM, XP, використовуються для управління IT-проектами розробки ПЗ. Методології впровадження інформаційних систем використовуються для організації проекту впровадження. Групові і індивідуальні практики (PSP, Personal Software process; TSP, Team Software process) служать для безперервного підвищення ефективності роботи команд і розробників.

В залежності від умов реалізації проекту розглядають прогнозовані та адаптивні.

Прогнозовані методології (рисунок 1.2) грунтуються на передумові про можливість і доцільність детального планування майбутнього. Для IT-проекту формулюються вимоги до системи, що розробляється, формується план проекту і визначаються потреба в ресурсах. Зміни в плані проекту і вимогах вважаються небажаними.

Рисунок 1.2 – Прогнозована методологія

Адаптивні методології націлені на подолання передбачуваної неповноти вимог, а також – їх постійної зміни. Прикладом адаптивних методологій є Scrum і Extreme Programming. Адаптивні методології (рисунок 1.3) враховують психологічні особливості процесу розробки ПЗ. Одним зі значимих чинників успіху використання адаптивних методологій є висока кваліфікація фахівців, в першу чергу – розробників.

Управлінські інструментальні засоби (software project management and process management) ґрунтуються на принципах теорії управління, в їх основі лежать такі концепції, як тотальне управління якістю, управління проектами, управління знаннями.

Рисунок 1.3 – Адаптивна методологія

Адаптивні методології націлені на подолання очікуваної неповноти вимог та їх постійної зміни. Коли міняються вимоги, команда розробників теж міняється. Команда, що бере участь в адаптивній розробці, майже не може передбачити майбутнє проекту [8]. Точний план існує лише на найближчий час. Більше віддалені в часі плани існують лише як декларації про цілі проекту, очікувані витрати і результати. Серед адаптивних методологій розрізняють

Scrum, Crystal, Extreme Programming, Adaptive Software Development, DSDM, Feature Driven Development, Lean software development (рисунок 1.4).

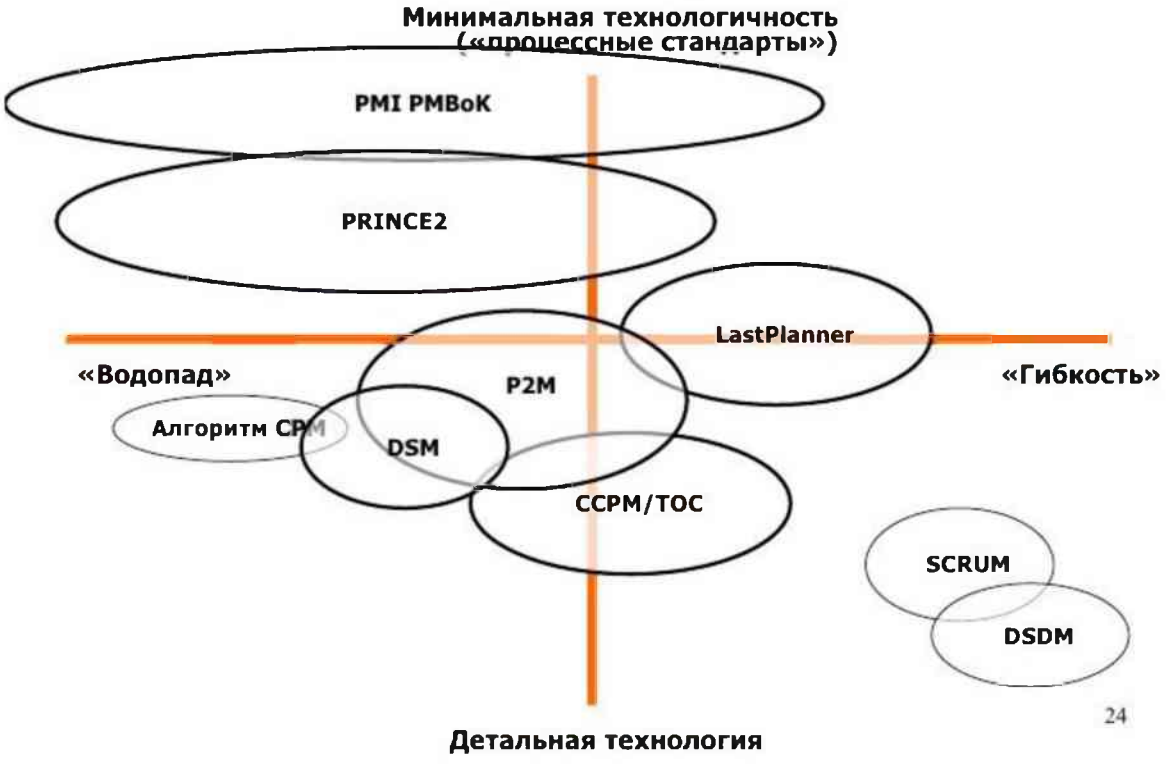

Рисунок 1.4 – «Болото» стандартів управління проектами

Управління проектами виникло як напрям в менеджменті, покликаний задовольнити потреби організацій в досягненні специфічних цілей в умовах високої складності і ризику.

В основі концепції управління проектами лежить поняття «проект».

Проект визначається як процес, намір, зусилля і дія. Під проектом розуміється комплекс робіт, спрямованих на досягнення запланованої і, як правило, унікальної мети. Проект включає проблему, засоби її вирішення та отримані в процесі реалізації результати (рисунок 1.5). Тому говорять про інваріантність визначення проекту.

Існуючі стандарти і методології визначають «проект» як:

– унікальний процес, що складається з набору взаємопов'язаних і контрольованих робіт з датами початку і закінчення, зроблений, щоб досягти мети відповідності конкретним вимогам, враховуючи обмеження за часом, витратам і ресурсам;

– процес досягнення поставленої мети у рамках особливого, конкретного комплексу умов.

– намір, який значною мірою характеризується неповторністю умов в їх сукупності, наприклад: завдання мети; тимчасові, фінансові, людські і інші обмеження; розмежування від інших намірів; специфічна для проекту організація його здійснення;

– дія, спрямована на створення унікального продукту або послуги.

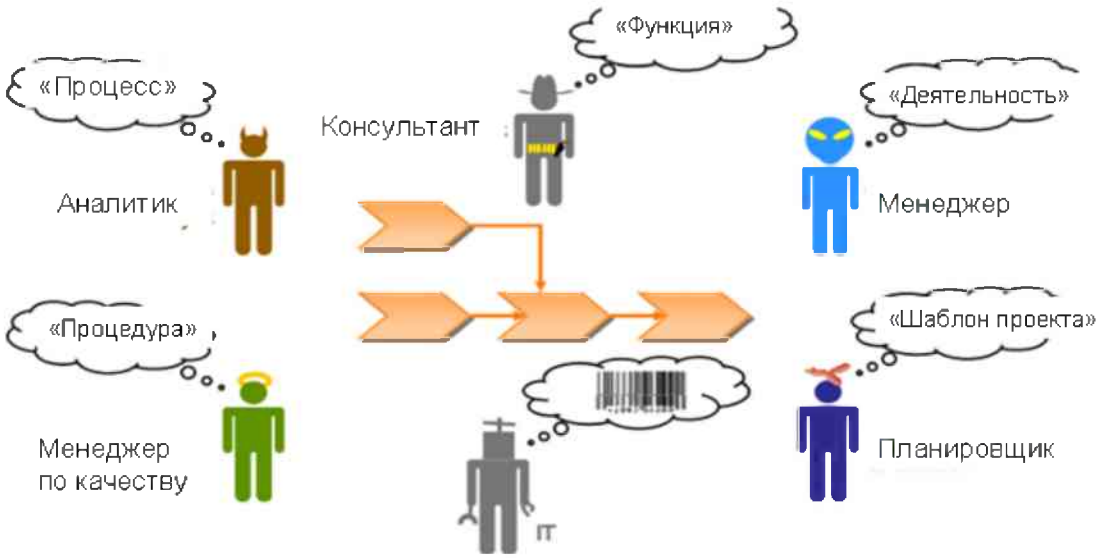

Рисунок 1.5 – Різниця в розумінні терміну «проект»

Управління проектом є визначенням, встановленням, регулюванням і розвитком зв'язків між елементами проекту, поставлених перед проектом цілей, що забезпечують досягнення. Опис кращих практик в області управління проектами представлені в PMBook – керівництві до зведення знань з управління проектами [9].

На рисунку (рисунок 1.6) представлені експертні області, необхідні для команди управління проектом.

Розробка програмного забезпечення є індустрією, в якій домінує проектна модель організації виробництва. Ідеї концепції управління проектами лежать в основі проектних методологій в сфері IT.

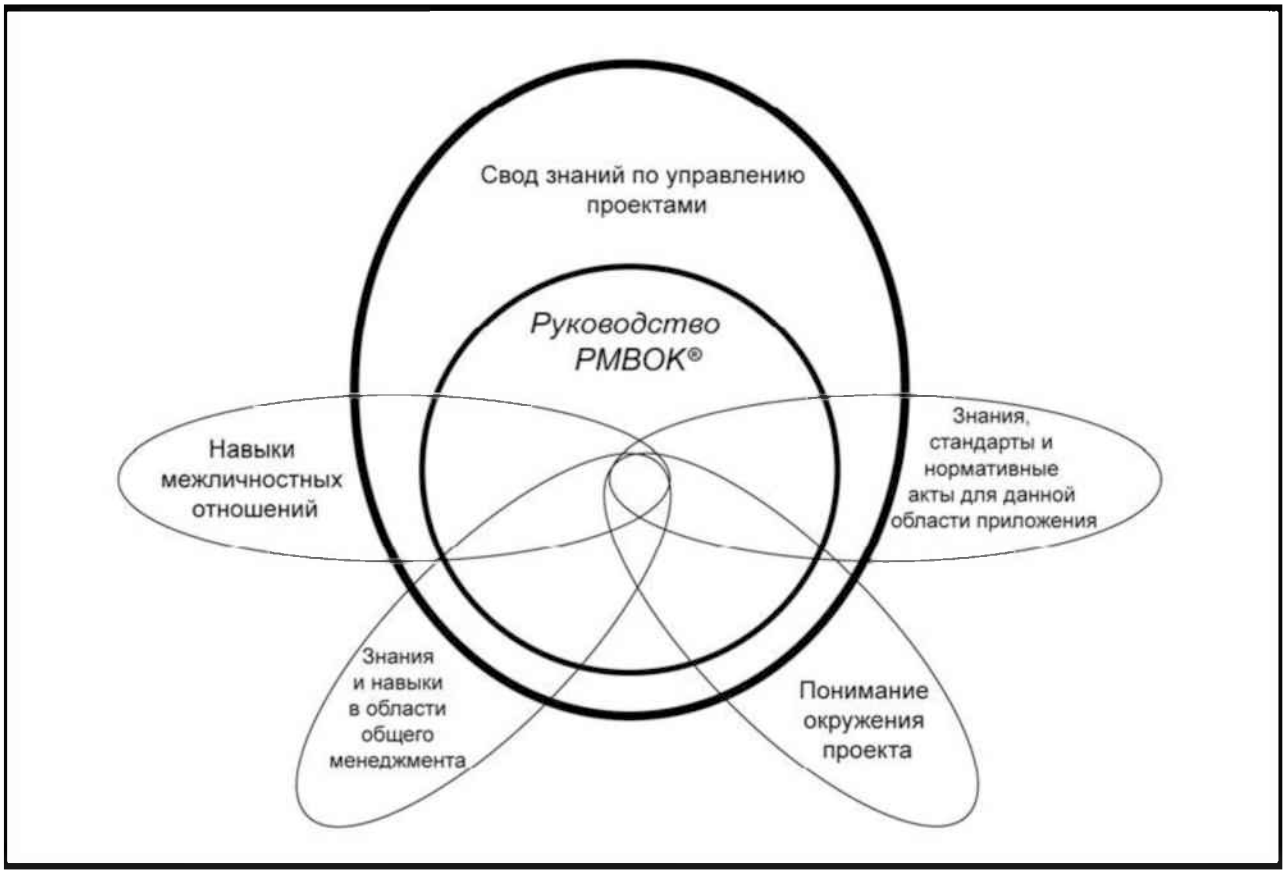

Рисунок 1.6 – Експертні області для команди управління проектом

До управління проектом входить:

– визначення вимог;

– установка чітких і досяжних цілей;

– урівноваження вимог, що суперечать, за якістю, змісту;

– часу і вартості;

– корекція характеристик, планів і підходу відповідно до думки і очікувань різних учасників проекту.

1.4 Аналіз умов застосування гнучких методологій розробки

1.4.1 Середовище застосування гнучкого методу управління проектом

Гнучкий метод розробки проводиться спільно з невеликою групою, що бажано розташовується в одній будівлі. Якщо група не може знаходитися фізично в одній будівлі, на допомогу приходять різні способи зв'язку і постановки завдань в режимі on-line (Skype, телеконференції, системи управління проектами). Роботи по програмуванню продукту виконуються упродовж серії сесій, де команда пише код, потім тестує робочі модулі системи, потім процес повторюється. При цьому рівень документації зведений до мінімуму, оскільки команда в основному покладається тільки на неформальне спілкування в межах команди. Подібна схема застосовується для інших груп учасників проекту(менеджери, маркетологи і тому подібне). Усе це відрізняється від традиційного підходу, де значна кількість часу приділяється плануванню і ведеться велика документація вимог. Команди, що застосовують гнучкий метод, визначають і дають пріоритети функціям, що розробляються, на основі їх цінності у бізнесі, а після того, як будуть створені критичні компоненти системи робота ведеться над тими моментами, які мають найбільш високий пріоритет [8], [10].

Компоненти гнучкого методу:

– візуальний контроль;

– високопродуктивні команди, розташовані поруч;

– розробка, що базується на тестах;

– управління, що адаптується;

– спільна розробка;

– розробка, що базується на елементах продукту, який розробляється;

– управління і співпраця в протилежність указам і контролю;

– переміщення фокусу з витрат на прибуток;

– засвоєння отриманих уроків.

1.4.2 Основні проблеми переходу на гнучкі методології

До основних проблем переходу на гнучкі методології можна віднести:

– нерозуміння ролі керівництва при впровадженні методології. Перехід на більшість гнучких методологій вимагає кардинальної зміни завдань і методів роботи керівників. Цей перехід можна сформулювати як перехід від управління до напряму, перехід від наказів і вказівок до рекомендацій. Перша помилка, яка допускається при такому переході – підсвідоме прагнення зберегти за собою владу, що повертає нас до вже звичного управління. Друга не менш серйозна помилка прямо протилежна – неправильне розуміння цього переходу може призвести до усунення менеджерів від керівних функцій, коли керівник перестає давати вказівки, але так і не починає давати рекомендації. Роль керівника при цьому зводиться до чисто формальних, секретарських функцій. Причиною цього може бути і неготовність керівників надавати аргументовану пораду (у старій моделі їм цього робити не доводилося) або побоювання невиконання висловлених рекомендацій [10];

– побудова «системи», що не має необхідної гнучкості. Відсутність досвіду роботи за новою методологією веде до того, що новий процес впроваджується за інструкціями, що веде до негнучкості і бюрократизації;

– початок впровадження не з «основи». При переході на нові методології керівництво переслідує конкретні вигоди, які повинна забезпечити нова методологія: висока продуктивність, якість і тому подібне. При цьому деякі практики нового процесу очевидніше ведуть до цих вигод, а деякі – зовсім не очевидно. У зв'язку з цим виникає спокуса впровадити спочатку «вигідніші» практики, а вже потім – всі інші. При цьому не враховується, що деякі практики є базовими по відношенню до інших, і впровадження других без перших неможливо;

– змінюються робочі місця, але не міняються звички. Проголошення нових правил і наслідування цих правил – це принципово різні речі. Недостатнє розуміння нових ідей усіма членами команди або слабке усвідомлення вигоди

від переходу на новий процес, слабка мотивація, відсутність перехідного періоду з явно вираженими послабленнями в графіку робіт і кількості завдань – це далеко не повний список помилок, здатних привести до того, що новий процес може прийматися лише формально, а насправді – саботуватися членами команди;

– все вимірювати, але ні на що не реагувати. Нескінченний аналіз ситуації замість безперервних поліпшень. Більшість гнучких методологій мають на увазі збір даних і обговорення помилок, що були допущені на попередньому етапі, перед початком наступного. Поширені помилки в цій сфері – збір даних без їх подальшого аналізу або невірна, занадто поверхнева інтерпретація зібраних даних;

– обходитися без підтримки. Відсутність досвіду роботи команди за новою методологією і впровадження по букві інструкцій містить багато помилок, невірних інтерпретацій і нерозуміння. Тільки участь в процесі переходу досвідченого інструктора, що має безпосередніх досвід роботи за новим процесом, може позбавити команду від безлічі невірних поворотів і безвиході або, в окремих випадках, навіть від побудови абсолютно невірної методології.

1.5 Постановка задачі

В рамках виконання кваліфікаційної роботи необхідно дослідити методи адаптивного управління програмним проектом, виявити їх особливості при використанні в IT-проектах, розробити алгоритм вибору гнучкої технології управління в залежності від умов її використання, запропонувати метод виявлення вузьких місць в організації проекту з використанням переспектив.

Як свідчить аналіз предметної галузі, більшість проблем, що виникають в гнучких проектах, обумовлена неточністю оцінки ефективності при розробці проекту. Вважається, що при використанні гнучкої методики проекти будуть займати менше часу, менше витрат і нести менше ризиків.

Постановка задачі полягає в створенні умов обґрунтованого вибору і адаптації методів адаптивного управління програмними проектами з урахуванням специфіки організації ІТ-підприємств.

Гнучки методології базуються на адаптації процесу розробоки до особливостей задач, які вирішуються під час розробки програмних продуктів, та продуктивності команди розробників [8]. Разом з тим формалізовані методи такої адаптації відсутні, що вимагає від дослідника побудови моделей, що адекватно відображають процеси розробки та шляхи їх вдосконалення протягом усього ЖЦ проекту.

Проблема формалізації або зображення шляхів виявлення технічних проблем в деякому вигляді, наприклад, в графічному, може бути актуальною для команд, які досить часто мають проблеми, пов'язані з прийняттям та переглядом управлінських рішень протягом проекту:

– розмір спринта може бути або переоцінено, або недооцінено;

– продуктивність усієї команди або її окремих членів може бути нижчим або вищим за очикуємий;

– конкретні задачі спринта можуть бути переоцінені, або недооцінені.

Такі проблеми досить часто виникають у стартап-команд, які лише починають формуватися, у разі проведення ребілдингу команди, яка ще ніколи не займалася написанням такого класу програмного забезпечення або під час перших спринтів, коли ще не визначена средня продуктивність команд.

Використання методів Process mining дозволяє побудувати модель процесу розробки ПЗ, яку у подальшому доцільно використовувати для аналізу під ча проведення ретроспективи. Технолігия аналізу процеса розробки полягає в структурному аналізі лога подій та фільтрації данных, побудова графічного відображення на підставі отриманих даних.

Таким чином, використання методів аналізу процесів дозволить забезпечити ефективну організацію проведення ретроспектив для визначення проблемних місць під час реалізації ІТ-проектів.

2 ПРИНЦИПИ ТА МЕТОДИ УПРАВЛІННЯ РОЗРОБКОЮ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Аналіз методологій, моделей і стандартів проведемо з використання виділених в попередньому розділі основних груп.

Проектні методології містять схожий набір елементів: опис моделі життєвого циклу розробки ПЗ, опис ролей проектної команди, опис артефактів, що розробляються, і рекомендації з використання допоміжних інструментальних засобів, таких як мови графічного модельована, метрики тощо. Основна відмінність між проектними методологіями пов'язана із застосуванням різних моделей ЖЦ ПЗ.

Методології управління IT-організацією, як правило, мають складнішу структуру. До цієї групи відносяться моделі зрілості (CMM, CMMI), стандарт оцінки можливості процесів (SPICE), індивідуальний процес розробки (PSP, TSP). Загальним для всіх принципом є наявність рекомендованих процесів, що не мають на меті досягнення безпосередніх цілей проектів розробки ПЗ, але необхідні для досягнення надійних і передбачуваних результатів у будь-якому з проектів [9], [10]. В основі всіх методологій цієї групи лежать ідеї TQM.

Доповнюють процесні методології оцінки процесів. Наприклад, для CMMI розроблена методології SCAMPI, що формалізує процедуру оцінки процесів. Модель IDEAL є методологією поліпшення процесів розробки ПЗ. Структура методологій цієї групи постійно розвивається, постійно йде пошук найбільш зрозумілого представлення рекомендацій. Наприклад, модель CMMI має два варіанти представлення: безперервне – continues і послідовне – staged.

2.1 Моделі життєвого циклу ПЗ за різними методологіями і стандартами

Моделі життєвого циклу ПЗ є центральною частиною проектних методологій. Вперше поняття «Модель життєвого циклу» ввів в широке використання Уїнстон Ройс в 1968 р. Він описав каскадну і водоспадну моделі

життєвого циклу розробки ПЗ, які є каркасом більшості проектів, виявив недоліки каскадної моделі і визначив, як вона може бути трансформована в ітеративну модель.

Модель життєвого циклу визначає склад і послідовність кроків по створенню і використанню ПЗ, що відбиває його різні стани, починаючи з виникнення необхідності і закінчуючи виходом із вживання: аналіз, проектування, кодування і тестування з вказівкою послідовності їх виконання, категорій виконавців, і використовуваних методів [9]. Методологія PRINCE розрізняє два поняття «product life span» і «project life cycle». Відповідно до цих позначень ЖЦ ПЗ охоплює лише частину життєвого циклу проекту, обмежену періодом розробки.

Розрізняють три формальні моделі ЖЦ ПЗ: поетапну з проміжним контролем (рисунок 2.1, а), каскадну (рисунок 2.1, б) та спіральну (рисунок 2.2).

а) б)

Рисунок 2.1 – Моделі ЖЦ ПЗ

Відповідно до каскадної моделі перехід на наступний етап може відбуватися тільки після завершення попереднього. Поетапна модель з проміжним контролем — це ітераційна модель із зворотними зв'язками між етапами.

Спіральна модель припускає циклічне виконання усіх етапів каскадної моделі, внаслідок чого та, що реалізовується технічних рішень перевіряється за допомогою прототипів. Кожен виток спіралі відповідає створенню фрагмента або версії ПЗ, на нім уточнюються цілі і характеристики проекту, визначається його якість і плануються роботи.

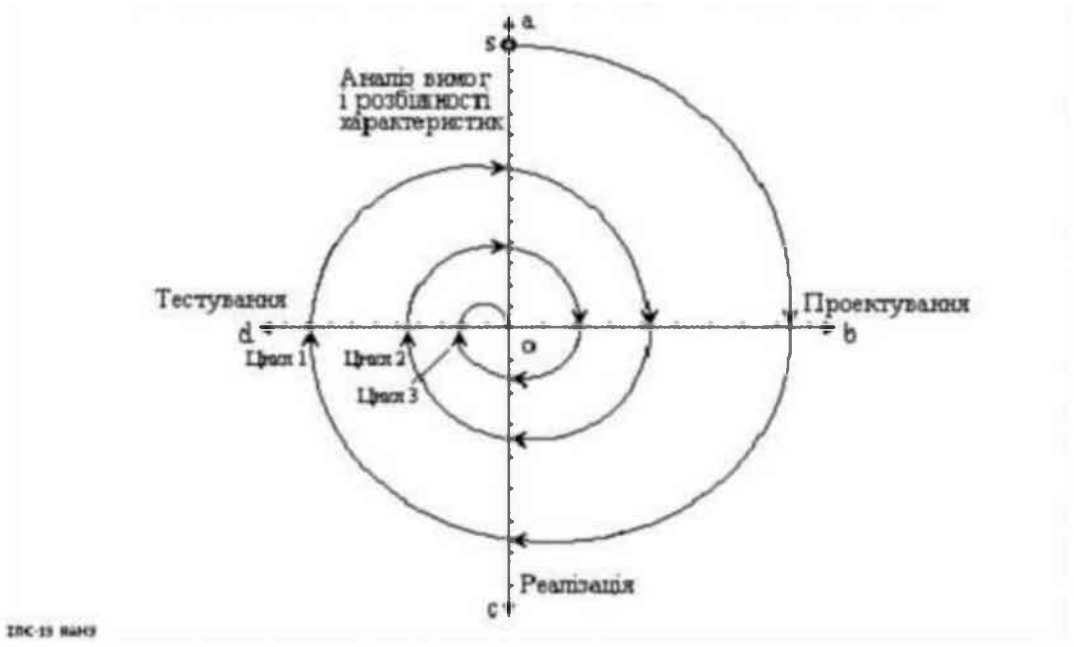

Рисунок 2.2 – Спіральна моделі ЖЦ ПЗ

Неповне завершення робіт на кожному з етапів дозволяє переходити на наступний етап, не чекаючи повного завершення роботи на поточному. При ітеративному способі розробки недоліки можна буде виправити на наступній ітерації. Головна задача цієї моделі – якнайшвидше показати користувачам системи працездатний продукт, тим самим активізуючи процес уточнення і доповнення вимог.

Модель процесу розробки є, за своєю суттю, описом ЖЦ ПЗ, якщо не враховувати допоміжні та організаційні процеси.

Крім розглянутих вище моделей ЖЦ ПЗ в якості самостійних моделей ЖЦ ПЗ виділяють прототипування та швидку розробку (RAD, Rapid Application Development).

Протипування припускає одночасну розробку функціональних вимог та топологічне проектування.

Швидка розробка базується на використанні прототипу, який будується на початку роботи над проектом [1]. При цьому усі вище вказані моделі ЖЦ ПЗ відносять до «прогнозуючих моделей», які ставлять оптимізацію вище адаптивності, включаючи і спіральну модель.

До групи адаптивних методологій відносять XP, SCRUM і Адаптивну розробку програмного забезпечення (Adaptive Software Development, ASD) [7].

Адаптивні методології грунтуються на ітеративній та інкрементальній розробці. Ітеративний процес виник практично одночасно з класичним послідовним підходом. В основі ітеративного підходу лежать концепції управління якістю, наприклад, орієнтована на підвищення якості методика Уолтера Шеварта з Bell Labs, що складається з серії коротких циклів кроків по плануванню, реалізації, вивченню і дії (plan - do - study - act, PDSA) (рисунок 2.3).

Існує декілька моделей і стандартів, що регламентують життєвий цикл. Більшість з них відносяться до замовленого ПЗ, та окрім безпосередньо ЖЦ ПЗ регламентують також і процеси розробки:

ISO/IEC 12207 : 1995 стандарт на процеси і організацію життєвого циклу. Поширюється на усі види замовленого ПЗ. Стандарт не містить опис фаз, стадій етапів. Модель життєвого циклу – це структура, що містить процеси, дії і завдання, які здійснюються під час розробки, функціонування і супроводу програмного продукту упродовж усього ЖЦ системи – від визначення вимог до завершення її використання.

Методика Oracle CDM (Custom Development Method) з розробки прикладних інформаційних систем під замовлення – конкретний матеріал, деталізований до рівня шаблонів проектних документів, розрахованих на пряме використання в проектах з опорою на інструментарій Oracle. Міра адаптивності CDM обмежується трьома моделями ЖЦ: класична (передбачені усі види робіт, завдання і етапи), «швидка» розробка (Fast Track), яка орієнтована на використання інструментів моделювання, і програмування Oracle, «полегшений»

підхід, рекомендований у разі малих проектів і можливості швидко прототипувати ПЗ.

Рисунок 2.3 – Ітеративна модель PDSA

Rational Unified Process (RUP) пропонує ітеративну модель розробки, що включає чотири фази: початок, дослідження, побудова і впровадження. Кожна фаза може бути розбита на етапи (ітерації), в результаті яких випускається версія для внутрішнього або зовнішнього використання. Проходження через чотири основні фази називають циклом розробки, кожен з циклів завершується генерацією версії системи. Якщо після цього робота над проектом не припиняється, то отриманий продукт продовжує розвиватися і знову проходить ті ж самі фази.

На рисунку 2.4 представлено узагальнену модель життєвого циклу продукту і проекту відповідно до моделі RUP.

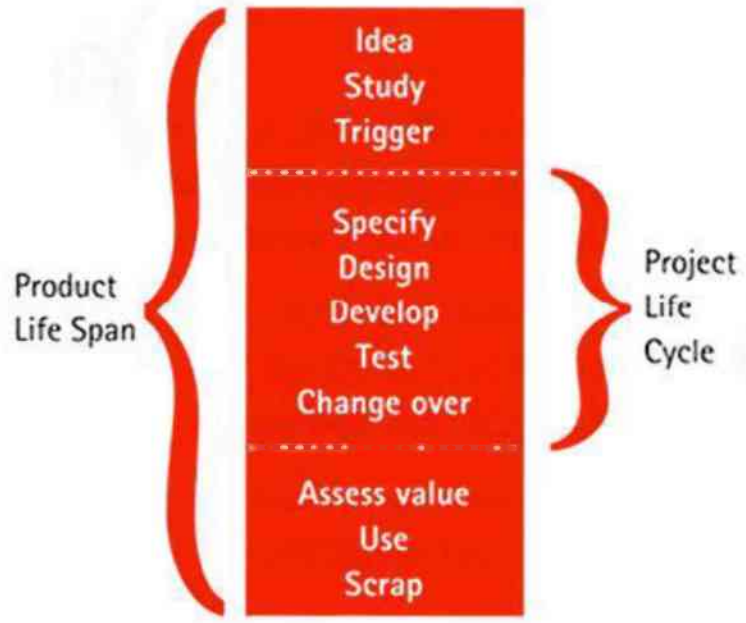

Рисунок 2.4 – Модель життєвого циклу продукту і проекту

Робота в рамках RUP – це створення і супровід моделей, а не паперових документів, тому цей процес прив'язаний до використання конкретних засобів моделювання (UML), а також до конкретної технології проектування і розробки.

Модель процесів MSF (рисунок 2.5) представляє собою узагальнену методологію розробки і впровадження IT-рішень. За рахунок власної гнучкості ця модель може використовуватись для розробки широкого кола IT-проектів, вона охоплює увесь життєвий цикл створення рішення, починаючи з найперших етапів до впровадженя. Модель процесів MSF поєднує у собі якості двох класичних моделей: каскадної та спиральної. Також модель MSF має багато спільного з моделлю RUP – вона так само включає чотири фази: аналіз, проектування, розробка, стабілізація, але додається ще один – розгортання.

У водоспадній моделі (рисунок 2.6) усі етапи виконуються послідовно один за іншим, при цьому наступний етап може початися тільки після завершення попереднього.

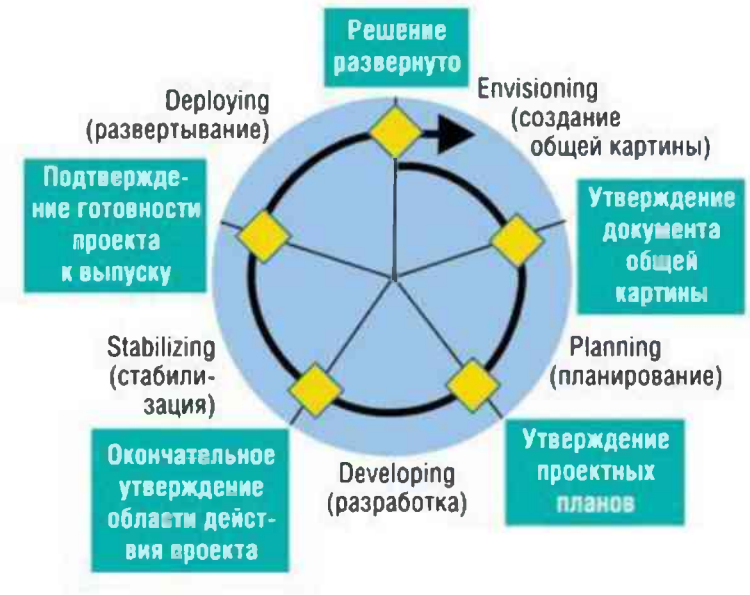

Рисунок 2.5 – Модель MSF процесу розробки

Рисунок 2.6 – Водоспадна модель життєвого циклу процесу розробки

Методика Oracle CDM (Custom Development Method) пропонує просту однорівневу модель процесів. Модель CDM представляє інтерес як зразок моделі процесів методології розробки бізнес-систем. Аналіз моделі показує високий рівень прагматизму. Модель прямо означає основні види робіт, які виконуються при проектуванні, розробки і впровадженні бізнес-систем. Наприклад, виділені

такі процеси, як дослідження існуючих систем(для вирішення питань інтеграції, апаратного забезпечення і так далі), процес перенесення даних. Саме фокус на інтеграції і даних відрізняє розробку бізнес-додатків.

CDM адаптована для розробки інформаційних систем для бізнесу. Процеси складаються з послідовностей завдань, завдання різних процесів взаємозв'язані явно вказаними посиланнями.

Раціональний уніфікований процес (Rational Unified Process, RUP) був розроблений Філіпом Кратченом за замовленням Rational Software. RUP пропонує одну з найбільш повних моделей процесів ітеративної розробки (рисунок 2.7).

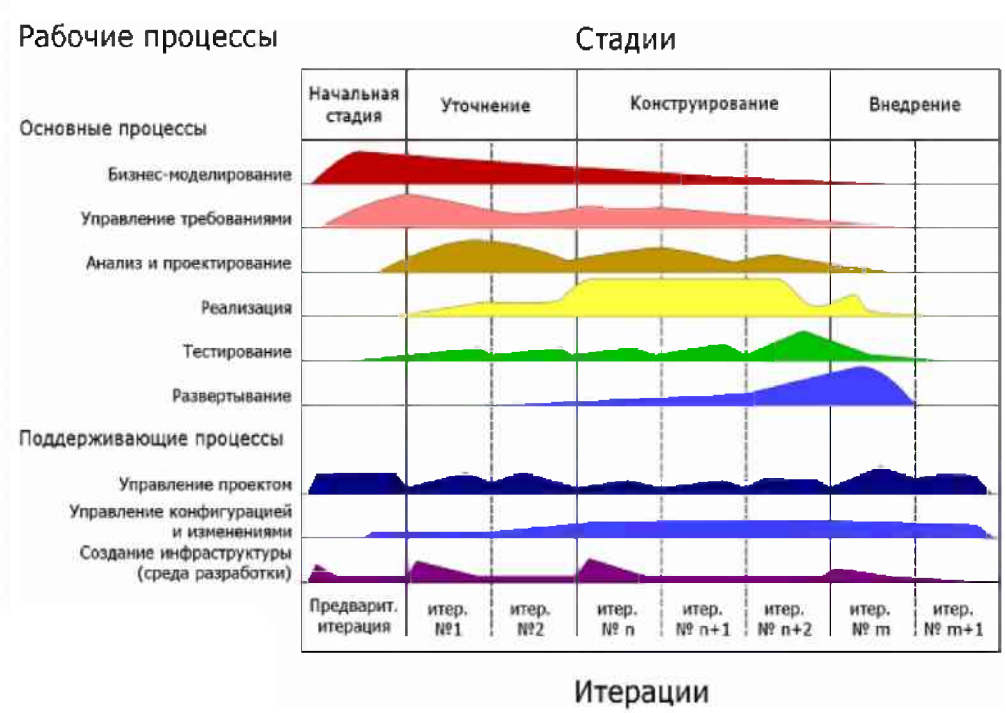

Рисунок 2.7 – Раціональний уніфікований процес, RUP

В середині кожного процесу зосереджені пов'язані між собою артефакти (програми і документи) і діяльності (завдання). На сьогодні RUP — це набір методів і практик для поліпшення якості і ефективності процесу розробки програм, а також базова архітектура і веб-інструментарій для обміну розробками

через Інтернет. Методологія і інструментарій були передані IBM співтовариству розробників з відкритим кодом безкоштовно в 2005 р.

RUP забезпечує строгий підхід до розподілу завдань і відповідальності усередині організації-розробника. Підвищення продуктивності колективної розробки досягається за рахунок використання накопиченого досвіду по створенню ПЗ, за допомогою керівництва, шаблонів і рекомендацій з використання інструментальних засобів для усіх критично важливих робіт, упродовж життєвого циклу створення і супроводу ПЗ.

Для застосування RUP необхідно використовувати спеціалізовані інструменти організації взаємодії між учасниками проекту. Надаючи кожному членові групи доступ до тієї ж самої бази знань, незалежно від того, чи розробляє він вимоги, проектує, виконує тестування або управляє проектом – RUP гарантує, що усі члени групи використовують спільну мову моделювання, процес, мають узгоджене бачення того, як створювати ПЗ. В якості мови моделювання використовується Unified Modeling Language (UML), що є міжнародним стандартом [3].

Особливістю RUP є те, що в результаті роботи над проектом створюються і удосконалюються моделі. Замість створення паперових документів, RUP спирається на розробку і розвиток семантично збагачених моделей, що усебічно представляють систему, що розробляється. RUP – це керівництво по тому, як ефективно використати UML. Стандартна мова моделювання, використовувана усіма членами групи, робить зрозумілими для усіх описи вимог, проектування і архітектуру системи.

RUP підтримується інструментальними засобами, які автоматизують великі розділи процесу. Вони використовуються для створення і вдосконалення різних проміжних продуктів на різних етапах процесу створення ПЗ, наприклад, при візуальному моделюванні, програмуванні, тестуванні тощо.

RUP – це процес, що конфігурується. Він придатний для використання як для невеликими групам розробників, так і для великих організацій, що займаються створенням ПЗ.

В рамках RUP робота над проектом здійснюється в термінах послідовності дій, продуктів діяльності, виконавців та інших статичних аспектів процесу з одного боку, і в термінах циклів, фаз, ітерацій і тимчасових відміток завершення певних етапів в створенні ПЗ, тобто в термінах динамічних аспектів процесу, з іншою.

Модель життєвого циклу об'єктної кластерної розробки. Технологія розробки впливає на вибір моделі ЖЦ ПЗ [11]. Найбільш популярною технологією розробки систем на сьогодні є об'єктно-орієнтоване проектування і програмування. Специфіка об'єктного дизайну (інкапсуляція, спадкоємство і поліморфізм) дозволяє прискорити розробку і істотно знизити ризики за рахунок паралельної роботи на кластерами – логічними групами класів.

Модель життєвого циклу об'єктної розробки ґрунтується на паралельній роботі над декількома кластерами, що використовують специфікації раніше завершених кластерів.

Етапи проектування і реалізації не мають чітких меж. Цей підхід, як і інші, вимагає проведення аналізу, на основі якого приймається рішення про вибір моделі ЖЦ ПЗ.

Краща за традиційні підходи, кластерна модель підвищує ефективність управління проектом шляхом гнучкого розподілу ресурсів. Щоб уникнути розбіжностей необхідно регулярно відстежувати поточні стани кластерів. Оптимальним є контроль керівником проекту ходу робіт протягом певного часу, наприклад, один раз на тиждень. Тим самим гарантується наявність на кожній стадії поточної демонстраційної версії, системи, яка не обов'язково охоплює усі аспекти, але готова для показу клієнтам. Крім того, це дозволяє своєчасно інформувати учасників проекту про зміни і усувати будь-яку неузгодженість між кластерами.

Можливість паралельної розробки в рамках кластерної моделі забезпечується за рахунок механізмів приховування інформації. Кластери можуть залежати один від одного, наприклад кластер графічного інтерфейсу може вимагати класи комунікаційного кластера для реалізації віддаленого

терміналу. Завдяки абстрактним даним можна працювати над кластером, навіть якщо кластери, від яких він залежить, ще не завершені. Ця можливість реалізується за наявності закінченої специфікації необхідних класів на основі їх офіційного інтерфейсу, заданого в короткій формі або у вигляді відкладеної версії.

Останній етап життєвого циклу кластера – узагальнення – не має аналогів в традиційних підходах. Мета – шліфовка класів для їх підготовки до повторного використання. Успіх політики повторного використання вимагає наявності певної культури повторного використання у кожного учасника, виділення достатніх ресурсів для розширення відповідних можливостей початкових версій класів.

Перевагою кластерної моделі є усунення різких переходів між окремими етапами, які використовують різні нотації, системи поглядів і персонал (аналітики, проектувальники, програмісти). Невідповідності між аналізом і проектуванням, проектуванням і реалізацією, реалізацією і розвитком є причиною зростання витрат і збільшення кількості помилок в традиційній схемі розробки ПЗ.

2.2 Структура проектних методологій

Практично усі проектні методології містять такі частини, як опис рекомендованої моделі ЖЦ ПЗ, модель команди і модель процесів. Опис моделей процесів містить опис робочих продуктів. Сучасні проектні методології містять опис умови їх ефективного використання, включаючи вимоги до характеристик проекту(об'єм проекту, характер ризиків, технології розробки, кваліфікація персоналу, а також географічне розташування команди). Крім того, проектні методології можуть доповнювати різні класифікації, наприклад, класифікації ризиків.

2.2.1. Методології швидкої розробки

Всі методології швидкої розробки ґрунтуються на принципах, сформульованих в Agile Manifesto. До методологій швидкої розробки відносяться методологія SCRUM, сімейство методологій Crystal та XP (рисунок 2.8).

Кен Швабер і Майкл Бидл запропонували методологію SCRUM [12]. В основі методології лежить цикл розробки. Цикл розробки в SCRUM складає 30 днів. Щодня в ході одного циклу проводяться 15 хвилинних нарад, які називаються SCRUM, робиться огляд зробленої з моменту проведення останньої наради, визначаються проблеми і розробляється план праць до наступної наради. В ході одного 30-денного циклу реалізуються вимоги Sprint Backlog, впорядковані у відповідність з пріоритетами, встановленими замовниками.

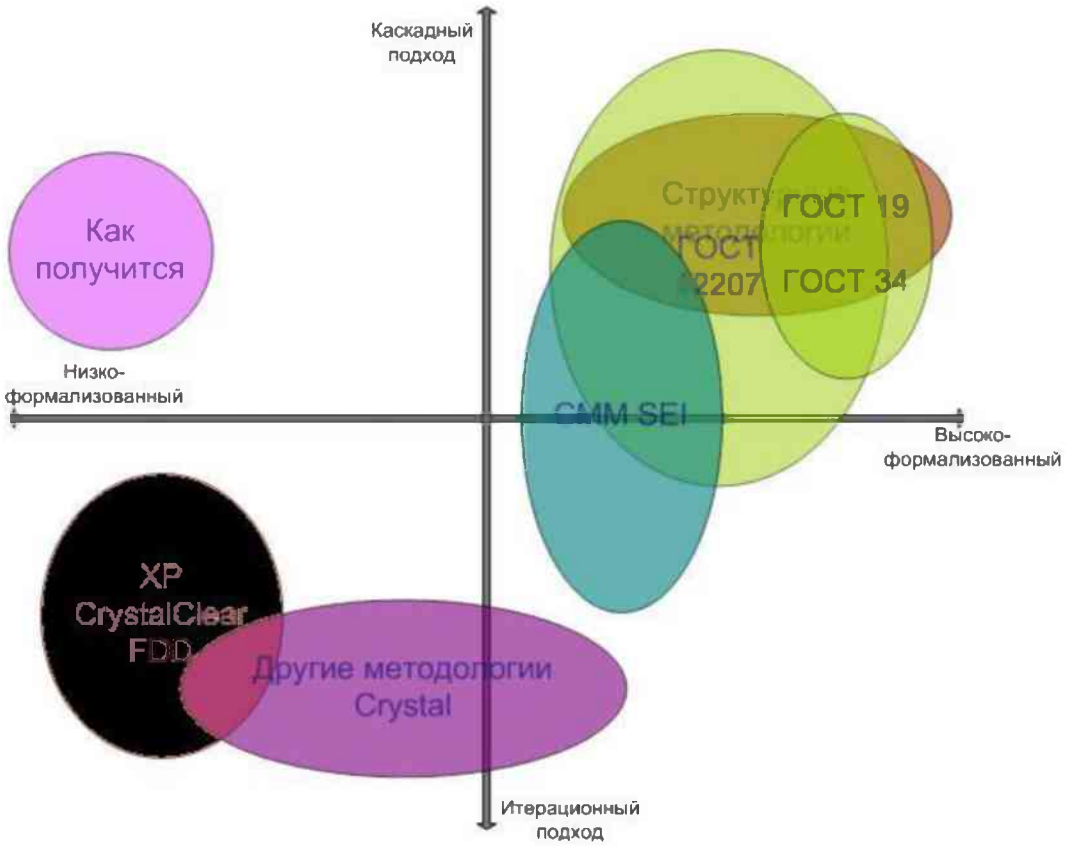

Рисунок 2.8 – Вибір методології

Алистер Коберн розробив сімейство методологій Crystal [2]. У сімейство входять чотири методології Crystal Clear, Crystal Yellow, Crystal Orange і Crystal Red. Методології відрізняються розміром команди розробників, кількістю різних ролей у складі команди.

Методології Crystal відрізняють наступні особливості:

– вони побудовані навколо людей і комунікацій;

– вони адаптуються, щоб відповідати параметрам: є набір компонентів, з якого команда проекту збирає методологію;

– розраховані на команди, розміщені в одному місці;

– існує можливість «розтягання» методології.

Методології Crystal можуть за необхідності запозичувати елементи XP, PSP та ін. В їх основі лежить інкрементна розробка. Тривалість інкременту – від 1 до 4 місяців. Робочі семінари проводяться регулярно: до, після і бажано – в середині інкременту.

Основною рисою методології є не дотримання практик, а підтримка комунікацій і досягнення двох цілей кооперативної гри – виконання проекту і реалізація наступного проекту.

Кент Бек формалізував методологію екстремального програмування. Методологія екстремальної розробки XP (eXtreme Programming) має найкоротший цикл розробки та малочислену команду. В методологи XP використовуються наступні поняття (рисунок 2.9):

– Спайк (Spike) – дуже проста програма, що розробляється, що б досліджувати можливі рішення;

– Призначені для користувача історії(User Stories) – варіанти використання системи, які пишуться замовникам;

– Приймальний тест (Acceptance Test) – варіант тестового прикладу, який розробляється для призначених для користувача історій. Тести мають бути автоматизовані, тому що виконуються дуже часто.

Разработка через тестирование (Test driven development - TDD)

Непрерывная интеграция (Continuous Integration)

Рефакторинг (Design Improvement, Refactoring)

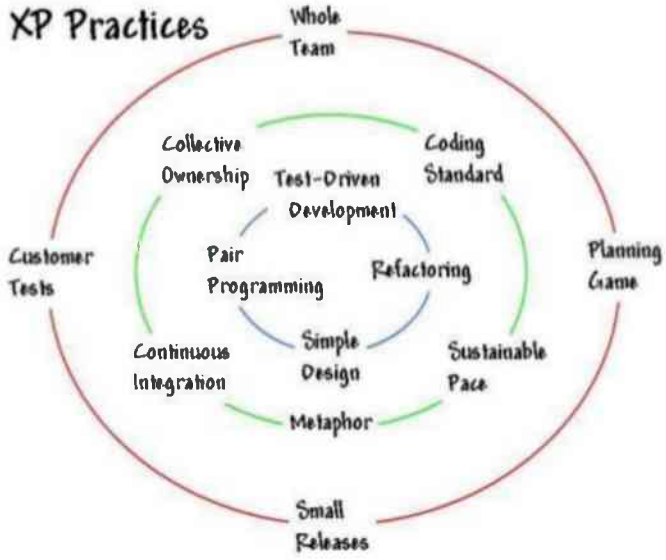

Рисунок 2.9 – Методологія eXtreme Programming

Відмінним елементом XP є обов'язковий рефакторинг – реінжиніринг коду, що періодично проводиться. Колективне володіння кодом і парне програмування також стали елементами методології XP.

XP є досить жорсткою з організаційної точки зору методологією, задає модель процесу розробки, склад артефактів, що розробляються, і структуру команди.

2.2.2. Система моделей Microsoft Solution Framework

MSF (Microsoft Solution Framework) було розроблено як частину загальної методології Microsoft – Microsoft Enterprise Framework, яка так само включає MRF (Microsoft Readiness Framework) і MOF (Microsoft Operations Framework). Зараз не передбачено сертифікацію компаній на відповідність MSF, система моделей використовується не як стандарт, а як керівництво, відповідаючи не на питання, «що робити», а надаючи конкретні рекомендації і відповідаючи на питання «як робити» (рисунок 2.9). MSF використовується на етапі проектування, розробки додатків та впровадження додатків. На момент

створення в 1994 році MSF була описом рішення проблем бізнесу за допомогою технічних засобів, поступово узагальнюючи кращу практику груп розробки, впровадження, клієнтів і партнерів Microsoft.

Рисунок 2.9 – Модель Microsoft Solution Framework

На відміну від інших методологій значна частина моделей MSF є описом конкретних прикладів розробки з коментарями.

2.2.3. Проектна методологія PRINCE

На відміну від методологій, розглянутих вище, фокусом методології PRINCE є не процес розробки ПЗ (ЖЦ ПЗ як його основа), а процес управління проектом, де розробка – це тільки його складова частина (рисунок 2.10). Істотна увага в PRINCE приділяється ініціації проекту і закриттю проекту. Методологія

PRINCE (PRojects IN Controlled Environments, проект в контрольованому середовищі) була розроблена у Великобританії в 1989 р. Central Computer and Telecommunications Agency(CCTA) як стандарт ведення IT-проектів. З часом межі використання цього стандарту розширилися, і сьогодні не обмежуються тільки сферою IT. Методологія PRINCE є розширенням концепції управління проектами. У 1996 р. вийшов стандарт PRINCE2, який став стандартом де-факто в управлінні проектами у Великобританії. Методологія має обмежену сферу застосування – вона описує вимоги до організації проекту розробки і впровадження інформаційних систем для бізнесу(комерційних застосувань).

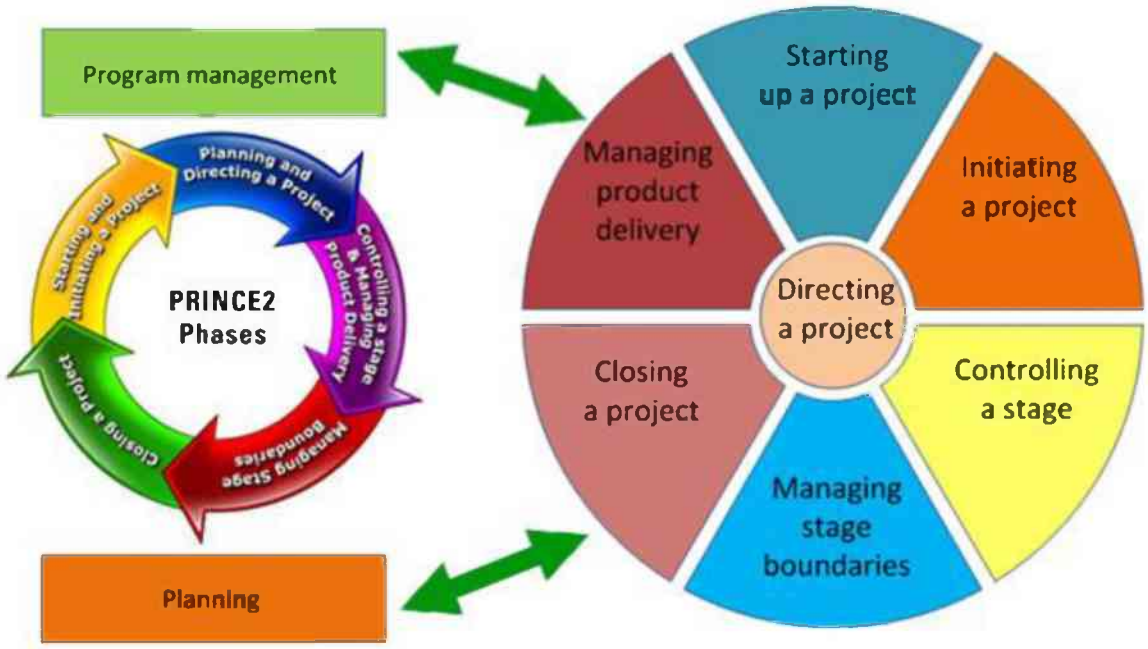

Рисунок 2.10 – Проектна методологія PRINCE

Проект спрямовується бізнес-сценарієм («business case»), який визначає інтереси організації(замовника), вимоги до продуктів, що розробляються, і інших результатів. План організовується навколо створюваних продуктів, сфокусований на отриманні результатів, а не виконанні певного набору робіт.

PRINCE не є заміною методологій швидкої розробки (SCRUM, Crystal і XP), вона може гармонійно доповнювати методології цього класу. MSF містить в собі практики, регламентовані PRINCE, ці методології є взаємозамінними.

2.3 Аналіз методів і інструментальних засобів отримання кількісних оцінок

Використання кількісних оцінок в управлінні розробкою ПЗ є необхідною умовою застосування підходів TQM і процесного управління. Використання статистичних методів для управління процесами розробки є показником рівня зрілості процесів, забезпечує стабільніші і надійніші характеристики планів, використовується для точнішої оцінки ризиків і забезпечує ефективне використання ресурсів. Статистичні методи лежать в основі безперервного управління якістю, надають необхідну для поліпшення процесів інформацію. Згідно концепції TQM, чисельне визначення цілей є необхідною умовою ефективного управління.

Для кількісного виміру характеристик процесів і продуктів у сфері розробки ПЗ використовуються «метрики». Існує формально визначення метрики як критерію, побудованого на основі двох або більше взаємозв'язаних заходів, в цьому випадку синонімом поняття «метрика» буде поняття «індикатор». Міра є результатом одиничного виміру деякого значення, що змінюється, наприклад розміру, часу роботи або кількості помилок. Наприклад, метрикою продуктивності є кількість функціональних точок (Functional Point, FP) за годину, а метрикою якості – кількість дефектів на тисячу рядків коду. В деяких випадках поняття «міра» (measure) і «метрика» (metrics) використовуються як взаємозамінні.

В таблиці 2.1 представлений структурований підхід до використання кількісних методів в управлінні розробкою ПЗ. В основі підходу лежить виділення об'єктів і категорій вимірів, на підставі чого формуються питання, які необхідно розв’язати.

Таблиця 2.1 – Об'єкти та категорії виміру метрик

Об'єкт	Категорія виміру	Питання
Графік проекту і оцінка міри виконання	Міра виконання окремих робіт	Чи буде проект завершений вчасно? Яка міра завершення окремих робіт?
Ресурси і витрати	Персонал	Чи проводилася робота відповідно до плану? Чи великі відхилення?
	Фінансові результати	Чи укладається проект в запланований бюджет?
	Доступні ресурси	Чи є в наявності необхідні ресурси?
Якість продукту	Помилки і дефекти	Чи може ПЗ бути поставлено покупцеві?
	Складність	Чи були вирішені попередні проблеми?
	Переписування коду	Чи являється ПЗ, що розробляється, тестопридатним і придатним для подальшої підтримки? Які зусилля витрачаються на внесення змін і виправлення помилок?
Збільшення розміру коду	Розмір продукту і незмінність	Чи змінюється розмір продукту і його функціональність?
Незмінність	Зрілість процесів	Чи змінюються вимоги до продукту?
Продуктивність	Продуктивність	Чи зможе розробник виконати вимоги з урахуванням існуючих обмежень? Яка вірогідність успішного завершення проекту (окремої роботи)? Чи відповідає продуктивність процесів (окремих етапів, конкретних співробітників) сучасним вимогам?
Техніка	Використання наявних комп'ютерів і іншого устаткування	Чи відповідає продуктивність наявних технічних засобів існуючим потребам?
	Вплив технології	Чи призвело застосування нової технології до тих результатам, які очікувалися?

Метрики дозволяють порівнювати роботу окремих співробітників, відділів, відстежувати хід виконання проектів. Метрики дозволяють порівнювати показники діяльності компанії з середніми значеннями в галузі (benchmarking). Консалтингові компанії і організації надають ці дані на комерційній основі. Таким чином, метрики використовуються як для ухвалення оперативних рішень в ході виконання проектів, так і стратегічного управління.

Стандарт ISO/IEC 25010:2011 [13] припускає використання зовнішніх (external) і внутрішніх (internal) метрик. Зовнішні метрики дозволяють оцінити характеристики ПЗ в процесі функціонування усередині системи. Системою може бути тестове або реальне середовище експлуатації ПЗ. Внутрішні метрики вимірюють статичні характеристики ПЗ (та інших продуктів: специфікацій, планів тестування, призначеної для користувача документації і документації розробки). Внутрішні метрики використовуються на ранніх стадіях розробки: дизайн, програмування, а зовнішні – на стадії тестування і експлуатації.

У основі класифікації метрик можуть лежати цілі, стадії (етапи) розробки, об'єкти виміру, технологія.

2.3.1 Метрики програмного коду

Підрахунок кількості рядків коду (SLOC, software lines of code) є однією з перших метрик, використаних для оцінки розміру програмного забезпечення. Ця метрика використовувалася для оцінки продуктивності праці програмістів, для оцінки складності програм, для прогнозування часу тестування. Перевагами цієї метрики є простота використання, зрозумілість, а недоліками - залежність від мови розробки, а так само негативний вплив на стиль програмування для набуття більш високих значень цієї метрики.

Не дивлячись на простоту метрики, необхідно уточнити правила підрахунку, визначити, чи враховуються рядки коментарів, порожні рядки. Як і для будь-якої іншої метрики необхідно забезпечити її послідовне застосування.

Очевидно, що метрика SLOC може використовуватися тільки для порівняння ПЗ, розробленого з використанням єдиної мови програмування, або різних мов, однакових, що мають, рівень абстракції і синтаксис. Навіть стандарти оформлення програмного коду роблять вплив на метрику SLOC. Для порівняння ПЗ, розробленого з використанням різних мов програмування, використовується альтернативна метрика MAELOC (Million assembly – equivalent lines of code) – кількісна метрика обсягу мільйону рядків коду в еквівалентних командах мови асемблера.

Об’єктні метрики. Об'єктно-орієнтований аналіз, проектування і розробка є лідируючим напрямом в програмній індустрії зараз, знаходяться на етапі зрілого розквіту, розроблені не лише теоретичні основи, але і ефективні інструментальні засоби об'єктного аналізу і разработки5. Об'єктно-орієнтований підхід полягає у формуванні набору об'єктів, властивих цій предметній області, внаслідок чого закладається фундамент для вирішення усіх завдань в цій області. Отже, для оцінки якості об'єктного коду або моделі необхідно використати спеціалізовані метрики.

Традиційні метрики припускають незалежну оцінку коду і структур даних, в той же час об'єктні метрики дозволяють одночасно оцінювати функціональність і дані як єдиний об'єкт. Об'єктні метрики використовують такі поняття, як методи, класи, зв'язування, спадкоємство. Метрики використовуються для оцінки внутрішньої структури об'єкту, його взаємодії з іншими сутностями, ефективності алгоритмів тощо.

Традиційні метрики також можуть використовуватися для оцінки об'єктно-орієнтованого ПЗ, наприклад, для оцінки методів об'єктів доцільно використовувати метрику складності, підрахунок кількості рядків коду, підрахунок коментарів.

2.3.2 Метрики обсягу виконаних роботи

Метрика функціональних точок (FP, Functional Points) використовується для оцінки необхідного часу розробки на ранніх етапах проекту(на етапі концептуального і логічного проектування). Для використання метрики функціональних точок досить мати вимоги до програмного забезпечення. Точність оцінки залежить від рівня деталізації вимог. Крім того, використовуючи термінологію FP можна погоджувати вимоги до програмного забезпечення. Ця метрика так само використовується для оцінки продуктивності праці розробників та обсягів робіт. FP може використовуватися без зміни на всьому протязі життєвого циклу проекту. Використання прозорого кількісного алгоритму для оцінки трудомісткості проекту робить конструктивнішим діалог між замовником ПО і розробником.

Існує значна кількість модифікацій методу функціональних точок: IFPUG, MARK II, Feature Points, Experience PRO. Відмінність між ними полягає у складі елементів, кількості рівнів складності і відповідній кількості функціональних точок, отже, для співставлення результатів оцінки, отриманих за допомогою різних методів, необхідні додаткові перетворення. MARK II, наприклад, пропонує підхід на основі логічних транзакцій (logical transactions), що складаються з входів, процесу і виходів, що забезпечує реальну незалежність від мов і засобів розробки. Разом з тим існують певні умови вибору тією або іншою модифікації методу FP залежно від призначення ПЗ. Наприклад, MARK II найбільш часто використовується для оцінки інформаційних бізнес-застосувань, орієнтованих на роботу з даними, але не враховує складність алгоритмів, яку необхідно враховувати для оцінки наукового і інженерного ПЗ; так само він не враховує специфічної складності систем реального часу.

Стандарт ISO/IEC 14143-6:2012 [14] містить концепцію методу функціональних точок і практика використання цього методу. Стандарт не містить опису конкретного методу, а визначає характеристики, які повинен мати будь-який метод, який може бути віднесений до методу функціональних точок.

Метод функціональних точок має велике концептуальне значення в теорії управління розробкою програмного забезпечення, оскільки цей метод є єдиною існуючою альтернативою метрик, що використовують кількість рядків коду, метод функціональних точок не залежить від мови розробки і може використовуватися на ранніх етапах проекту.

До недоліків методу слід віднести значну складність використання, обумовлену необхідністю додаткового навчання співробітників використанню спеціалізованих засобів збору метрик. Крім того, на відміну від підрахунку рядків коду, який може бути виконаний з використанням автоматичних засобів, метод функціональних точок ґрунтується на експертних оцінках складності, отже, точність оцінок залежить від кваліфікації експертів і підвищуватиметься з часом у міру зростання кваліфікації експертів.

На точність оцінок впливає якість специфікацій і адекватність їх відображення в конкретних технічних рішеннях, оскільки метод використовує не самі функціональні вимоги, а характеристики проектного рішення.

Таким чином, використання цього методу може викликати зміни в процесі розробки ПЗ: якщо до використання методу на етапі контрактних переговорів використовувалися тільки функціональних вимоги, то після переходу до оцінки тривалості проекту і об'єму робіт з використанням функціональних точок знадобиться попереднє проектування.

Очевидно, що суворі вимоги до точності оцінок, які висуваються на ранніх етапах розробки, вимагають значних витрат на початку проекту. Існує певна двоїстість у використанні метрик. З одного боку, використання методу функціональних точок дозволяє отримати більш точні оцінки та уникнути ризику недооцінки складності проекту. Також цей метод побічно призводить до кращої деталізації специфікації, що зменшує ризик невиконання суттєвих вимог.

Але, з іншого боку, використання метрик збільшує витрати, у тому числі і витрати, які фірма несе до укладення контракту, що також збільшує фінансові ризики.

2.3.3 Метрики складності

Існують два аспекти оцінки складності: оцінка складності процесу і оцінка складності об'єктів розробки. Інформаційна складність програм характеризується кількістю різних типів і структур даних, що подаються на вхід та є результатом роботи програми.

Програмні модулі (функції, процедури, методи класів) є простими компонентами, тому вони доступні для кількісного аналізу. Програмне забезпечення є сукупністю програмних компонент, тому його складність характеризується кількістю зв'язків між ними та складністю взаємодії. Вимір складності потрібний не лише для оцінки потреби в ресурсах, але і для планування тестування, оскільки відомо, що між складністю і кількістю помилок існує пряма залежність.

Поширеною метрикою складності є метрика цикломатичної складності МакКейба (McCabe). Метрика складності дозволяє оцінити потребу в тестуванні і документуванні [13]. Існуюча практика припускає оцінку кількості помилок і тривалості тестування впродовж процедури тестування на основі існуючого потоку виявлюваних помилок. Основними недоліками цього підходу є неможливість передбачити тривалість тестування до того, як станеться істотне зниження інтенсивності потоку помилок, а так само прогнозувати потік помилок на стадії експлуатації системи. При цьому тестові приклади будуються у відповідність до специфікацій. Поширеним методом тестування є також структурне тестування, при якому виконуються усі гілки алгоритмів. З використанням поняття цикломатичної складності можна скласти ефективніший план тестування, при якому мінімальна кількість тестів забезпечує перевірку усіх ділянок коду.

Крім того, може існувати обмеження на рівень складності, в цьому випадку модулі, що мають складність вищу за встановлений рівень повинні бути перепроектовані для їх спрощення. Причиною для такого підходу є істотне зростання потенційної кількості помилок в коді, який досягнув певної

складності. Існують правила перетворення алгоритмів, які дозволяють зменшити цикломатичну складність.

Окрім оцінки складності логіки необхідно так само оцінювати її якість. З метою забезпечення легкості підтримки коду (maintainability) слід мінімізувати кількість управляючих структур. Метрика внутрішньої складності (essence complexity metrics) дозволяє оцінити рівень структурованості коду. Оцінка складності програмної системи (наприклад, інформаційної системи) не може бути отримана безпосередньо з оцінок складності окремих складових.

2.3.4 Метрики вартості

Однією з відомих моделей вартості проекту є COCOMO (Constructive Cost Model) – модель прогнозування витрат на розробку, вперше опублікована Барри Боемом (Barry Boehm) в 1981 році. Модель було оновлено в 1998 році, в ній було враховано повторне використання коду, об'єктно-орієнтований підхід, а також зрілість процесів розробки (COCOMO II).

COCOMO II [15] є моделлю використання сукупності різних метрик для прогнозування вартості розробки. Модель припускає використання альтернативних способів оцінки розміру проекту, наприклад підрахунок рядків коду і метод функціональних точок. У разі використання функціональних точок модель надає можливість переведення результатів цієї метрики в SLOC з використанням таблиці перетворень для обраної мови програмування.

Крім того, модель враховує, що на ранніх етапах проекту може бути відсутня інформація про платформу, розробників, необхідну функціональність. Модель враховує точність оцінки розміру програмного забезпечення на різних етапах проекту залежно від повноти інформації, пропонується статистика по проектах, що показує прогнозоване і фактичне значення тривалості проекту. Враховуючи той факт, що на ранніх етапах інформації недостатньо, модель пропонує використати укрупнені показники витрат на початку проекту, переходячи до детальних показників на наступних стадіях. Пропонуються три

моделі прогнозування вартості: Модель прототипу (Application Composition Model), Модель етапу проектування (Early Design) та Модель детальної розробки (Post Architecture).

Модель прототипу призначена для створення прототипів додатка на самих ранніх етапах розробки. Модель етапу проектування припускає використання методу функціональних точок і семи чинників витрат. Модель детальної розробки ґрунтується на вибраній моделі життєвого циклу, враховує можливі ризики розробки. Для оцінки розміру може застосовуватися підрахунок рядків коду або метод функціональних точок, з урахуванням повторного використання. Модель детальної розробки використовує сімнадцять показників витрат та п'ять показників масштабування проекту.

У моделі враховується ефект від повторного використання. Мінімальні витрати на використання без модифікації (витрати на оцінку, вибір, порівняння) виявляються на 5% у більшості випадків. Навіть невеликі зміни викликають значне зростання витрат, пов'язаних з необхідністю вивчення ПЗ, яке необхідно замінити, а також – з необхідністю аналізу інтерфейсів.

Витрати зростають від 5% до 55% при необхідності зміни до 10% коду, потім відбувається зниження зростання витрат і при збільшенні числа необхідних змін до 70% витрати складаю приблизно 70% вартості і далі ростуть практично лінійно.

2.4 Інтелектуальний аналіз процесів розробки інформаційних систем

2.4.1 Поняття Process mining

Process Mining або інтелектуальний аналіз процесів (ІАП) – це підхід, який дозволяє аналізувати бізнес-процеси на підставі журналів подій інформаційних систем. Основна мета Process mining полягає в використанні даних про події з метою отримання інформації про деякі бізнес-процеси, якими супроводжується

розробка ПЗ [16]. Слід зазначити, що дана технологія базується на технології інтелектуального аналізу даних (ІАД).

Process mining дозволяє в суворій відповідності до ЖЦ ПЗ більш наочно відслідковувати хід бізнес-процесів під час розробки ПЗ, можливо виявити приховані зв'язки між об'єктами бізнес-процесів, що дозволяє значно підвищити актуальність і якість моделі.

Модель бізнес-процесу, що супроводжує розробку ПЗ, є ядром функціонування процесно-орієнтованої системи. На теперішній час поширеним підходом до побудови процесно-орієнтованих систем є її розробка у відповідності до концепції Business Process Management (BPM). Концепція BPM включає методи, технології та засоби підтримки розробки, впровадження, управління та аналізу бізнес-процесів, що використовуються під час розробки ПЗ [17].

2.4.2 Технологія Process mining в рамках BPM

Життєвий цикл описує різні етапи управління конкретного бізнес-процесу (рисунок 2.11). На етапі проектування розробляється модель бізнес-процесу. Ця модель перетворюються в працездатну систему на етапі її конфігурації / реалізації. Якщо модель вже використовуються та WFM або BPM системи вже запущені, цей етап може бути дуже коротким. Проте, якщо модель носить неформальний характер і повинна бути перетворена в програмне забезпечення, ця фаза може зайняти значний термін часу.

Після того як система починає підтримувати розроблені процеси, починається етап впровадження / моніторингу. На цьому етапі запускаються бізнес-процеси, за якими проводиться моніторинг, щоб переконатися, чи потрібні будь-які доповнення / зміни в моделі бізнес-процесу. Деякі з цих доповнень / змін обробляються на етапі налаштування моделі.

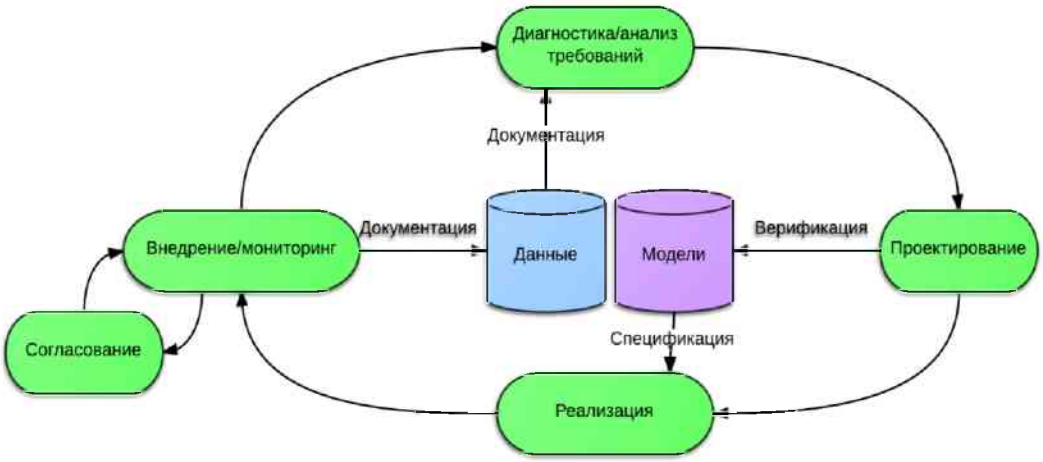

Рисунок 2.11 – Технологія Process mining

На цьому етапі в процес не вносять ніяких змін і не створюють нове програмне забезпечення – вносяться зміни / коригування в окремі елементи управління для адаптації або переналагодження процесу.

На етапі діагностики / визначення вимог оцінюється процес в цілому, а також здійснюється моніторинг виникаючих потреб у зв'язку зі змінами у навколишньому середовищі процесу. Низька продуктивність (наприклад, нездатність виконувати вимоги сервісів або недостатня підготовленість виконавця) або зміни і додавання нових вимог можуть викликати нову ітерацію життєвого циклу BPM починаючи з фази проектування [10].

Модель процесу відіграє основну роль в моделюванні і конфігурації / реалізації фаз розробки, в той час як дані відіграють свою роль лише у моніторингу та діагностики фаз. Також на рисунку 2.11 показано різні варіанти використання моделі процесу.

2.4.3 Методи Process mining

За відправну точку Process mining приймають журнал подій. Всі методи Process mining (рисунок 2.12) припускають, що існує можливість послідовно

записувати події, і кожна з таких подій належить до деякої активності (тобто до чітко визначеного кроку в процесі), що пов'язана з окремим випадком (тобто з екземпляром процесу) [17].

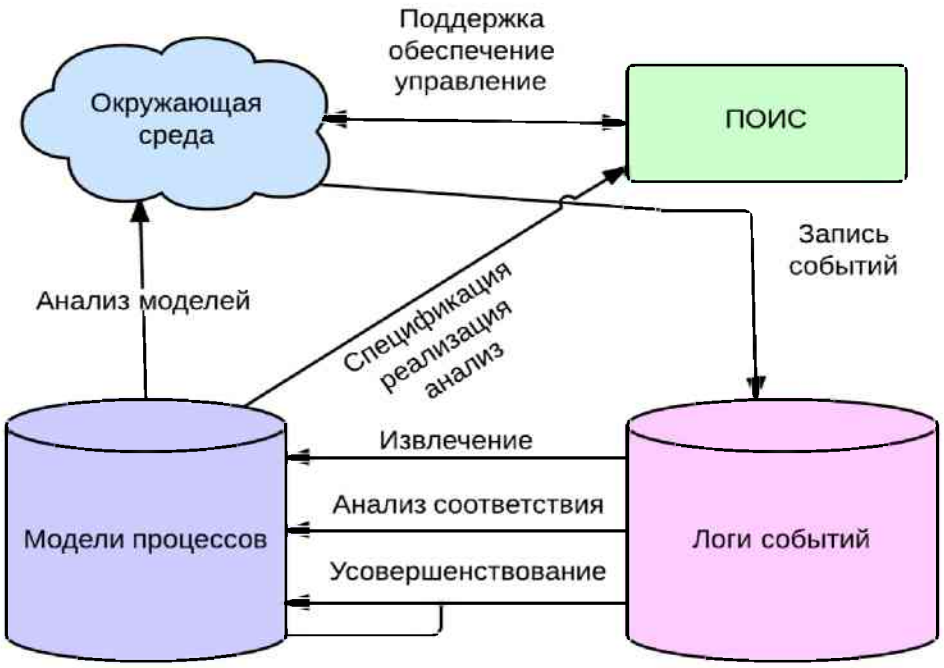

Рисунок 2.12 – Взаємодія між компонентами Process Mining

Методи Process mining в тих випадках, коли це можливо, використовують додаткову інформацію, що зберігається у журналі, про ресурси, виконання або ініціалізацію активності, час події або додаткові дані про подію.

Як видно з рисунку 2.12, журнал подій використовують в трьох типах Process mining:

– виявлення процесів (Process discovery) – це найбільш відомий вид Process mining. Метод будує модель «як є» на підставі наявного логу подій. На вхід метод приймає записи в журналі подій та генерує модель без використання будь-якої заздалегідь підготовленої інформації;

– перевірка відповідності (Conformance checking). При використанні даного методу існуюча модель процесу порівнюється з базовою моделлю того ж

самого процесу. Метод дозволяє побачити, наскільки реальні дані, що записані в журналі подій, відповідають моделі або навпаки;

– покращення (Enhancement). Ідея методу полягає в тому, щоб розширити або покращити існуючу модель, використовуючи інформацію про поточний процес, яка записана в журналі подій. Якщо метод відповідності перевіряє відмінності між моделлю і реальним процесом, то метод покращення є корисним при зміні або розширенні існуючої моделі.

Типи Process mining представлено на рисунку 2.13.

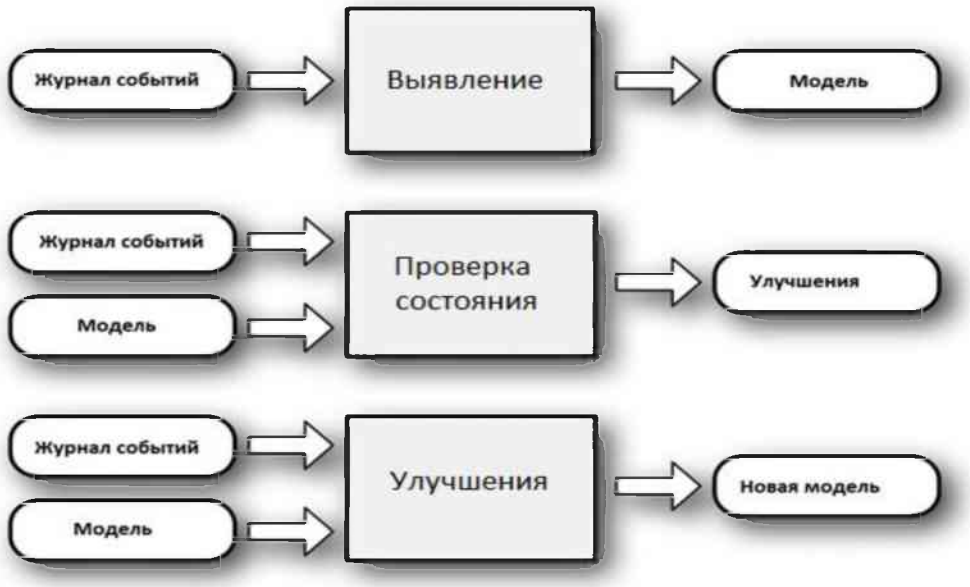

Рисунок 2.3 – Типи Process mining

2.4.4 Особливості проведення Process mining

У загальному вигляді проведення аналізу бізнес-процесів засобами Process mining характеризується наступними особливостями:

– аналіз процесів не зводиться просто до виявлення шляху. Фокус може бути зорієнтованим не на послідовності, а на часу проходження, організації і ресурсах, виборі типового кейса (примірника процесу) тощо;

– аналіз процесів не зводиться до окремого випадку аналізу даних, бо в ньому є притаманна процесам паралельність і точки вибору, які ігноруються більшістю алгоритмів з Data mining, тобто потрібні інші типи уявлень і інші алгоритми, які враховують паралельність та розгалуження;

– аналіз процесів не обмежується аналізом журналів подій. Наприклад, час завершення незакінченого оформлення клієнтського замовлення може бути передбачено на основі виявленої моделі процесу в реальному часі;

– журнали подій бувають різної якості, тому під час розробки ПЗ необхідно приділяти увагу не лише алгоритмам, а й логам.

2.5 Алгоритм аналіз файлів журналу засобами Process mining

На теперішній час існує значна кількість процесно-орієнтованих систем, які використовують свої правила організації ЖЦ бізнес-процесів. Досить часто аналіз функціонування таких систем скрутний в силу особливостей архітектури та організації реалізації тих чи інших бізнес-функцій. Загальною властивістю всіх цих систем є те, що вони всі ведуть реєстрацію подій за допомогою журнальних файлів.

Джерелом інформації для технологій аналізу процесів є журнал подій інформаційних систем. Однак не всі протоколи подій підходять для аналізу процесів – необхідно щоб в ньому була присутня інформація, якої достатньо для застосування методів ІАП.

Визначимо вимоги до інформації, яка повинна бути присутня в журналах:

– всі події, які записані в протоколі, повинні бути ідентифіковані з екземплярами процесів;

– всі події повинні бути впорядковані за часом їх виконання;

– різнотипні події повинні відрізнятися.

На підставі проведеного аналізу виділяють структуру журнальних файлів для систем управління проектами, що містять такі загальні атрибути:

– тимчасова мітка;

– завдання (ідентифікатор завдання);

– стан події;

– користувач (код користувача);

– додаткові атрибути дії (складність, витрачений час, пріоритет тощо).

Відповідно до структури журналів реєстрації подій в системі [20], кожен запис характеризується наступною множиною параметрів:

$$S_A =< T, A, O >, \quad (2.1)$$

де S_A –файл журналу;

T – параметр часу;

A – дія (подія);

O – об’єкт процесу, над яким виконується дія.

У загальному вигляді для журнальних файлів характерні наступні особливості:

– зареєстровані події відображають елементарні дії в інформаційній системі в рамках деякого процесу;

– зареєстровані події мають мітку про час їх виконання;

– зареєстровані події пов'язані з виконавцями та / або деякими ресурсами.

Використання методів аналізу журнальних подій систем контролю розробки програмних продуктів дає наступні переваги:

– існує можливість отримати результати про виконання процесів (конкретних завдань), що базуються на реальних фактах, відображених в журналі реєстрації подій;

– існує можливість побудувати чернетку моделі бізнес-процесів або уточнити існуючу;

– існує можливість виявлення потенціалу для оптимізації та визначення «вузьких місць» в бізнес-процеса;

– на підставі отриманих результатів можна організувати ефективний контроль за виконанням бізнес-процесів;

– отримані результати можна використовувати в подальшому для прогнозування / планування.

Для закріплення вимог і уніфікації протоколів, які оброблюються алгоритмами Process mining, було запропоновано стандарт записи протоколів MXML. MXML – це розширюваний формат, який заснований на мові розмітки XML. Він використовується для представлення та зберігання інформації у вигляді логів подій. Формат фокусується на ключовій інформації, яка необхідна для використання методів ІАД [17].

Кореневим вузлом кожного MXML-документа є WorkflowLog, що представляє лог-файл. Кожен WorkflowLog може містити довільну кількість вузлів Process. Кожен елемент типу Process є групою подій, які відбулися протягом виконання будь-якого процесу. Одноразові виконання цього процесу представлені елементами типу ProcessInstance. Таким чином, кожен ProcessInstance фіксує одноразове протікання процесу. Кожен ProcessInstance містить групу з довільної кількості елементів типу AuditTrailEntry (контрольних записів), кожна з яких відповідає унікальній події в логу. Кожний контрольний запис повинен містити як мінімум два елементи: WorkflowModelElement (назва завдання, яке було виконано) та EventType (тип події, який описує стадію виконання завдання). Формат також підтримує два необов'язкових поля, які, тим не менш, часто зустрічаються: Timestamp містить точну дату і час, коли відбулася подія, а Originator ідентифікує ресурс, тобто людину або інформаційну систему, які були ініціатором події. Розширюване поле Data може містити довільну кількість атрибутів, які складають пари рядків <назва-значення>.

Структуру MXML-формату наведено на рисунку 2.14 у вигляді діаграми класів.

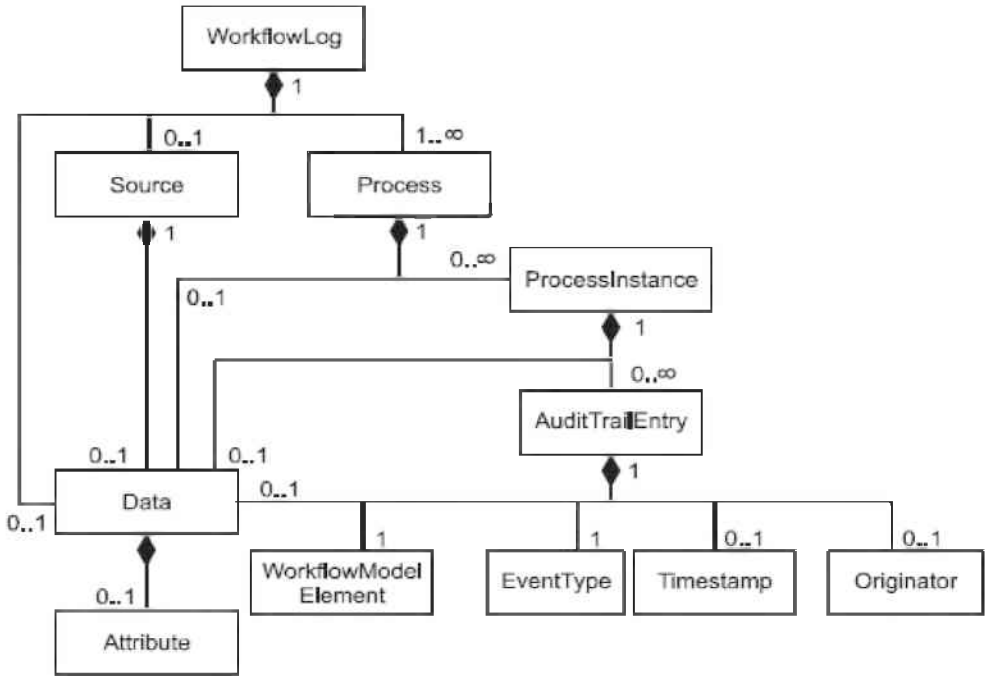

Рисунок 2.14 – UML-діаграма MXML-формату

3 ОПТИМІЗАЦІЯ ВИБОРУ І АДАПТАЦІЯ МЕТОДІВ УПРАВЛІННЯ ПРОЦЕСАМИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Організація процесів вибору методів управління

Розв’язання задачі вибору пов’язане з рішенням ряду технічних, організаційних і концептуальних завдань. Для вирішення завдання вибору оптимальної методології необхідно:

– сформувати розуміння можливості і необхідності вибору;

– забезпечити формування бібліотеки моделей, стандартів, техніки;

Об'єкти зберігання мають бути класифіковані, визначено їх призначення, сферу застосування, зв'язок з іншими елементами. Мають бути розроблені процедури оновлення архіву, що забезпечують отримання нових версій матеріалів і включення нових матеріалів. Практично неможливо сформувати абсолютно повну і актуальну базу матеріалів, для цього знадобилися б занадто великі зусилля.

– забезпечити процес постійної самоідентифікації компанії, включаючи визначення її культури, маркетингової стратегії, політику в області якості;

Необхідно однозначно сформулювати ті принципи і переконання, які лежать в основі вибору методології. (Наприклад, у вигляді тез-тверджень: тільки повне визначення усіх вимог дозволить успішно завершити проект, технічний фахівець може добре володіти тільки одним програмним засобом тощо).

– забезпечити постійне навчання і перенавчання співробітників компанії, в першу чергу – менеджерів проектів і фахівців відділу якості.

Таким чином, необхідно створити та безперервно покращувати наступні процеси:

– навчання співробітників інструментальним засобам;

– формування бібліотеки інструментальних засобів;

– обирання проектних та процесних інструментальних засобів (з використанням принципів верифікації);

– адаптації проектних та процесних інструментальних засобів (з урахуванням обліку специфіки проекту, вимог клієнтів, бізнес-мети організації тощо);

– оцінки існуючої практики використання інструментальних засобів (оцінка процесів, оцінка проектів).

Адекватний вибір моделі процесів є передумовою ефективного управління на основі процесного підходу. Вибираючи модель процесів, і, отже, певний стандарт, необхідно враховувати цілі та існуючі обмеження. Наприклад, якщо робимо лише перші спроби формалізації процесу, то можна скористатися моделлю SPICE для порівняння існуючої практики та рекомендацій стандарту. Модель процесів ISO 12207 слід використовувати для узгодження стадій та етапів ЖЦ ПЗ між клієнтом і виконавцем. Модель процесів CDM дозволить сконцентруватися на специфіці розробці інформаційних систем для бізнесу, проектуванні структури даних, взаємодії із замовником та таке інше.

Жоден з процесів не може бути використаний для усіх випадків розробки ПЗ. Практичний підхід припускає, що необхідно мати можливість адаптації процесу для широкого спектру додатків. Процес повинен забезпечувати економію при великих масштабах проекту і повернення інвестицій за рахунок використання загальної ідеї самого процесу, широкої автоматизації, загальних архітектурних зразків і компонентів.

Порівняльний аналіз теоретичних і практичних моделей процесів свідчить, що основні відмінності проявляються на рівні виділення процесів і структур, тоді як кінцевий склад робіт практично співпадає.

Основною функцією менеджменту є ухвалення рішень. Специфіка управління розробкою ПЗ визначає наявність пограничної області рішень, яка не може бути однозначно класифікована як чисто технічна або управлінська – наприклад, рішення, яку технологію використати для розробки, якими засобами користуватися для управління версіями або автоматизації тестування. Більшість таких рішень повинна прийматися в два етапи: на першому визначається набір

допустимих рішень, і оцінюються технічні можливості, потім обирається остаточне рішення з урахуванням стратегії компанії та маркетингових прогнозів.

Використання референтних моделей процесів дозволяє істотно знизити трудовитрати і підвищити якість моделювання бізнес-процесів. Цей підхід добре зарекомендував себе при впровадженні інформаційних систем на промислових підприємствах.

Успішність використання стандартів і моделей залежить від реалістичності початкових очікувань, які з ними зв'язувалися. Нереалістичними є очікування, що формальне дотримання методології управління IT-проектом та виконання вимог стандартів дозволяє отримати високу якість програмного продукту шляхом, незалежним від кваліфікації програмістів і інших фахівців, дозволити повністю уникнути залежності проекту від конкретних виконавців. Високий рівень зрілості дозволяє зменшити ризики втрати або спотворення інформації, підвищити ефективність роботи.

3.2 Особливості реалізації гнучкої методології Agile

Сам по собі Agile не є методом управління проектами. Це скоріше набір ідей і принципів того, як потрібно реалізовувати проекти. Методологія Agile – це сімейство процесів розробки, а не єдиний підхід до розробки програмного забезпечення [18, 19]. Вже на основі цих принципів були розроблені окремі гнучкі методи або, як їх іноді називають, фреймворки (frameworks): Scrum, Kanban, і багато інших. Ці методи можуть досить сильно відрізнятися один від одного, але вони слідують одним і тим же принципам.

Найголовніший плюс Agile – його гнучкість і адаптивність. Він може підлаштуватися практично під будь-які умови і процеси організації. Саме це зумовлює його нинішню популярність і те, скільки систем з різних сфер було створено на його основі. Один із принципів Agile: «Реакція на зміни важливіше, ніж виконання плану» [7]. Саме швидка і відносно безболісна реакція на зміни є

причиною того, що багато великих компаній прагнуть зробити свої процеси більш гнучкими.

Цінності та принципи Agile методології закріплені в документі «Agile Manifesto». Agile не включає конкретних практик, а визначає цінності і принципи, якими керуються успішні команди. Маніфест підписали представники наступних методологій [18]:

– Extreme programming;

– Scrum;

– Adaptive Software Development;

– Crystal Clear;

– Feature-Driven Development;

– Pragmatic Programming.

Більшість гнучких методологій націлена на мінімізацію ризиків шляхом зведення розробки до серії коротких циклів, званих ітераціями, які зазвичай тривають один-два тижні. Кожна ітерація сама по собі виглядає як програмний проект в мініатюрі і включає всі завдання, необхідні для видачі прості по функціональності: планування, аналіз вимог, проектування, кодування, тестування і документування.

Хоча окрема ітерація, як правило, недостатня для випуску нової версії продукту, мається на увазі, що гнучкий програмний проект готовий до випуску в кінці кожної ітерації. Після закінчення кожної ітерації команда виконує переоцінку пріоритетів розробки.

Слабка сторона полягає в тому, що кожній команді доведеться самостійно складати свою систему управління, керуючись принципами Agile. Це непростий і тривалий процес, який потребує змін всієї організації, починаючи процедурами і закінчуючи базовими цінностями. Він потребує також серйозних адміністративних ресурсів та витрат.

Agile-методи роблять акцент на безпосереднє спілкування. Принципи Agile методології:

– задоволення клієнта за рахунок ранньої та безперебійної поставки замовленого ПЗ;

– швидке урахування зміни вимог, навіть наприкінці розробки (це може підвищити конкурентоспроможність отриманого продукту);

– надійна поставка робочого ПЗ (кожен місяць або тиждень або ще частіше);

– тісне, щоденне спілкування замовника з розробниками протягом всього ЖЦ проекту;

– проектом займаються мотивовані особистості, які забезпечені потрібними умовами роботи, підтримкою і довірою;

– рекомендований метод передачі інформації – особисте спілкування;

– працююче ПЗ – кращий вимірювач прогресу;

– спонсори, розробники і користувачі повинні мати можливість підтримувати постійний темп на невизначений термін;

– необхідно постійну увагу приділяють поліпшенню технічної майстерності і зручному дизайну;

Існують методології, які дотримуються цінностей і принципів, заявлених в Agile Manifesto. До них відносяться фреймворк Scrum і метод Kanban.

Однією з найбільш поширених є методологія розробки Scrum, яку можна вважати набором конкретних практик, які використовуються в процесі розробки ПЗ [20].

Гнучкий фреймворк Scrum, створений в 1986 році, вважається самим структурованим з сімейства Agile. Він поєднує в собі елементи класичного процесу і ідеї гнучкого підходу до управління проектами. Слідуючи концепції Agile, Scrum розбиває проект на частини, які відразу можуть бути використані замовником для отримання цінності, так звані «беклоги» (product backlog). Потім ці частини проходять процес пріоритетності. Найважливіші елементи першими відбираються для виконання в спринті - так називаються ітерації в Scrum, що тривають від 2 до 4 тижнів. В кінці спринту замовнику видається робочий інкремент продукту - ті найважливіші елементи, які вже можна використовувати.

Наприклад, сайт з частиною функціоналу або програма, яка вже працює, нехай і частково. Після цього команда проекту приступає до наступного спринту. Тривалість у спринту фіксована, але команда вибирає її самостійно на початку проекту, виходячи з тематики проекту і власної продуктивності [5].

В рамках методології Scrum на кожен спринт виділяється наступний склад команди спринту (рисунок 3.1):

– Scrum Майстер.

– Замовник / Власник продукту.

– Команда.

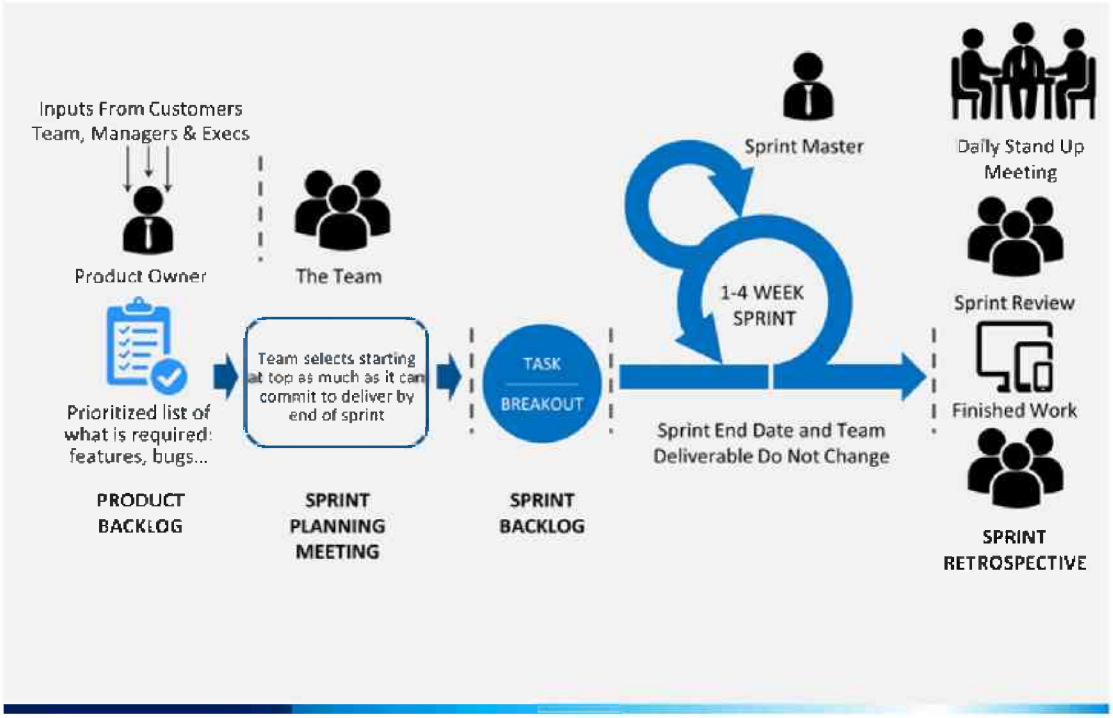

Рисунок 3.1 – Процес розробки ПЗ за методологією Scrum

Розглянемо кожну групу учасників більш детально.

Scrum Master – одна з головних ролей в методології. Scrum Master відповідає за успіх усього Scrum (як принципу організації роботи команди) в проекті. По суті, Scrum Master є інтерфейсом між менеджментом і командою. Як

правило, цю роль у проекті відіграє менеджер проекту. Основними обов'язками Scrum Master є:

– створює атмосферу довіри;

– усуває перешкоди;

– робить проблеми і відкриті питання видимими;

- відповідає за дотримання практик і процесу в команді.

Scrum Master веде щоденні збори команди спринту (Daily Scrum meeting) і відстежує прогрес команди за допомогою Списку завдань спринту, відзначаючи статус всіх завдань у спринті. Scrum Master може також допомагати Замовнику у створенні списку завдань спринту для команди.

Замовник / Власник продукту – це людина, яка відповідає за розробку продукту. Як правило, це менеджер продукту продуктової розробки, менеджер проекту для внутрішньої розробки і представник замовника для розробки на замовлення. Замовник – це єдина точка прийняття остаточних рішень для команди в проекті, саме тому це завжди одна людина, а не група або комітет. Обов'язки Замовника такі:

– відповідає за формування бачення продукту;

– управляє рентабельністю;

– управляє очікуваннями замовників і всіх зацікавлених осіб;

– координує і розставляє пріоритети у списку завдань спринту;

– надає зрозумілі і придатні для тестування вимоги команді;

– відповідає за приймання коду в кінці кожної ітерації.

Замовник ставить завдання команді, але він не має права ставити завдання конкретному членові проектної команди протягом спринту.

Команда. У методології Scrum команда самоорганізується і самокерується. Команда бере на себе зобов'язання по виконанню обсягу робіт на спринт перед власником продукту. Робота команди оцінюється як робота єдиної групи. У Scrum внесок окремих членів проектної команди не оцінюється, оскільки це розвалює самоорганізацію команди. Обов'язки команди такі:

– відповідає за оцінку елементів, що плануються для імплементації;

– приймає рішення по дизайну і імплементації;

– розробляє програмне забезпечення і передає його замовнику.

Основна структура процесів Scrum (рисунок 3.2) обертається навколо 5 основних компонентів: упорядкування беклога, планування спринту, щоденні наради, підведення підсумків спринту та ретроспективи спринту.

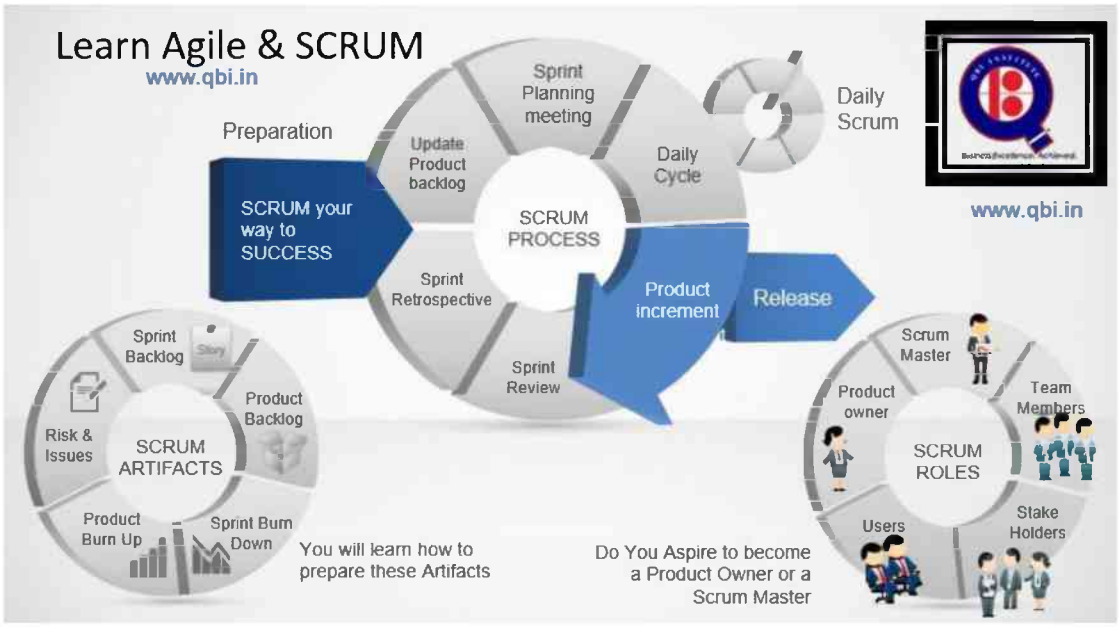

Рисунок 3.2 – Структура процесів Scrum

Щоб переконатися в тому, що проект відповідає вимогам замовника, які мають властивість змінюватися з часом, перед початком кожного спринту відбувається переоцінка ще не виконаного змісту проекту та внесення в нього змін. У цьому процесі беруть участь всі - члени команди, Scrum Master та власник продукту. Команда відповідає за те, щоб в кінці спринта всі необхідні завдання були зроблені, а поставки – виконані.

Серед сильних сторін варто відзначити те, що Scrum був розроблений для проектів, в яких необхідні «швидкі перемоги» у поєднанні готовності до змін. Крім того, цей фреймворк підходить для ситуацій, коли не всі члени команди мають достатній досвід у сфері, в якій реалізується проект - постійні комунікації

між членами команди дозволяють ліквідувати брак досвіду або кваліфікації одних співробітників за рахунок інформації та допомоги від колег.

Kanban дуже схожий на схему промислового виробництва [21]. Тобто, інкремент продукту передається вперед з етапу на етап, а в кінці виходить готовий до постачання елемент. Крім того, творець Kanban надихався супермаркетами, а саме їх принципом – «тримай на полицях тільки те, що потрібно клієнту». А тому в Kanban дозволяється залишити незакінчену задачу на одному з етапів, якщо її пріоритет змінився і є інші термінові завдання. Невідредагована стаття для блогу, підвішені без дати публікації або частина коду функції, яку можливо не включатимуть в продукт - все це нормально для роботи системи Kanban (рисунок 3.3). Kanban менш жорстикй, ніж Scrum – він не обмежує час спринтів, немає ролей, за винятком власника продукту. Kanban навіть дозволяє члену команди виконувати кілька завдань одночасно, чого не дозволяє Scrum.

Рисунок 3.3 – Процес функціонування Kanban

Для роботи з Kanban необхідно визначити етапи потоку операцій (workflow). У Kanban вони зображуються як стовпці, а завдання позначають спеціальні картки. Картка переміщається по етапах, подібно деталі на заводі, що переходить від верстата до верстата, і на кожному етапі відсоток завершення стає вище [4]. На виході ми отримуємо готовий до постачання замовнику елемент продукту.

3.3 Розробка алгоритму вибору гнучкої методології

Більшість успішних проектів – це ті, які слідували гнучким принципам розробки. Методи, засновані на традиційних моделях, сьогодні не завжди є найкращими, зокрема, при управлінні змінами, швидкої реалізації проекту або навіть задоволенні потреб ринку. Оскільки методи гнучкої розробки досить нові, тому лише деякі IT-компанії починають активно використовувати його.

Керівники проектів з часом усвідомили, що багато проектів зазнавали краху через строгі вимоги, неправильне планування, нездатність команди адаптуватися до змін. Здебільшого, вимоги клієнтів і користувачів змінювалися протягом циклу розробки, таким чином, до моменту випуску застосунку IT-продукт значно відрізнявся від того, що було заплановано. Крім того, до кінця процесу розробки було витрачено понад план часових і фінансових ресурсів. Тому проектні менеджери стали переходити на гнучкі методи управління.

Досить часто з різноманіття інструментів складно вибрати саме той, який найкраще підходить для проекту. Кожен проект унікальний, тому важко описати все різноманіття проектів. Однак, щоб почати застосовувати той чи інший інструмент гнучкого управління, необхідно зрозуміти – чи готова компанія працювати за гнучкою схемою. На основі принципів і впливу Agile на організацію були розроблені основні критерії готовності організації прийняти вказану методологію:

– ступінь формалізації – параметр охоплює бюрократію в проекті. Якщо в проекті необхідно заповнювати велику кількість паперів, виконувати багато різноманітних регламентів, то проект менше підходить для гнучкого управління.

– ступінь складності внесення змін – якщо ринок швидко змінюється і компанія може приймати такі швидкі рішення щодо змін розробки продукту, то можна вважати, що проекту підходить принцип гнучкого управління.

– системне рішення проблем – команда застосовує спеціальні методи вирішення проблем або є показники для вибору рішення, то такі дії належать до Agile.

– ступінь колективної відповідальності – команда бере участь на всіх етапах проекту та існує прозорість в організації проектної діяльності. Гнучкі методи також залежать від менеджера проекту, команди і типу проекту, тобто, як організований процес, і яким буде ІТ-продукт.

Дерево рішень алгоритму зображено на рисунку 3.4.

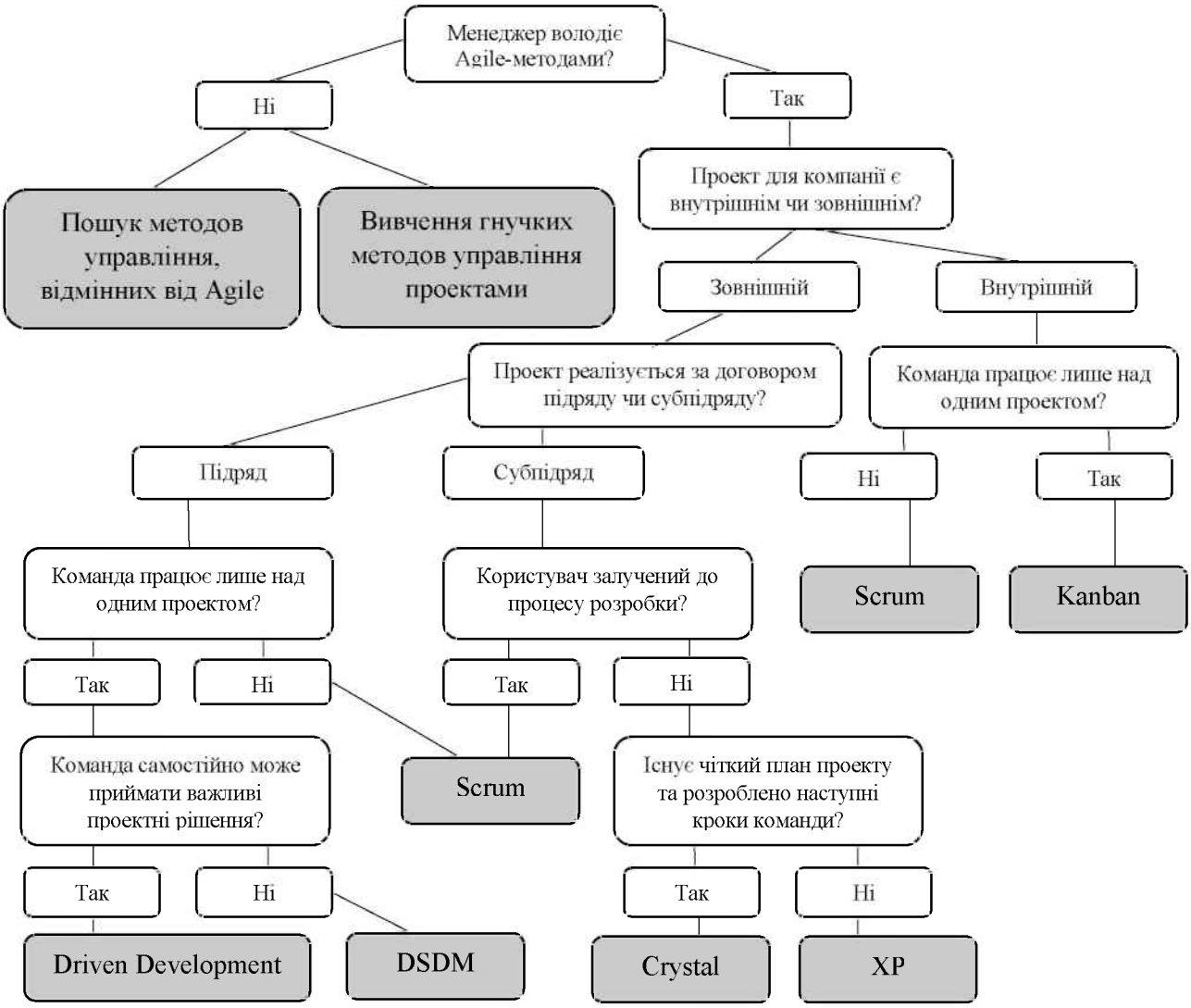

Рисунок 3.4 – Алгоритм вибору інструменту управління проектом

Обравши конкретну методологію розробки ПЗ, не слід повністю її дотримуватись. Досить часто її необхідно доопрацьовувати, адаптувати під конкретний проект. Щось з базової методології можна виключити, а щось, навпаки, можна добавити з інших методологій або ввести щось своє. Наприклад,

звичайною практикою вважається парне програмування у водоспадній методології, якщо це принесе користь проекту. Так само існує значна кількість прикладів одночасного застосування Scrum та Kanban в одному проекті.

Таким чином, на підставі проведеного аналізу предметної області, визначення властивостей проектів та особливостей управління ними було запропоновано алгоритм вибору гнучкої технології. Слід зазначити, що незважаючи на запропонований алгоритм, який значно полегшує вибір методології, остаточний вибір методології для управління проектом повинна робити особа, що приймає рішення, виходячи з власного розуміння ситуації та пріоритетів.

3.4 Побудова моделі процесу розробки програмного продукту

Як вже було зазначено вище, у якості вихідних даних для методів Process mining виступають журнальні файли (логи) інформаційних систем. Проблема вхідних даних полягає в тому, що для аналізу підходить не кожен з журнальних файлів: журнал подій повинен мати формат, придатний для опрацювання методами Process mining – MXML. Додаток ProM, який досить часто використовують для проведення аналізу бізнес-процесів, використовує саме цей формат.

Для проведення ефективного аналізу виконання процесу розробки необхідно обмежити дані журнального файлу, скоротивши його до необхідного обсягу. Оскільки на ретроспективах обговорюються проблеми і переваги минулого спринту, необхідно обмежити дані з урахуванням часу проведення спринту. Це дозволяє бачити динаміку процесу виконання завдань в рамках конкретного проміжку часу.

Також під час проведення ретроспективи інколи виникає потреба в більш детальному аналізі прогресу виконання конкретних завдань якоїсь активності конкретного виконавця [22]. З цією метою доцільно обмежити загальний обсяг

даних журнального файлу до певного числа конкретних подій, що стосуються певного користувача або завдання.

Визначимо алгоритм застосування методу Process mining для побудови моделі наступним чином.

Крок 1: трансформація вихідних даних з метою побудови моделі процесу розробки методами Process mining в необхідний формат.

Крок 2: модифікація назв завдань з урахуванням їх статусу і виконавця.

Крок 3: формування обмежень за розміром спринту, виконавцями, функціональністюж

Крок 4: формування вихідних даних для побудови моделі з урахуванням наявних обмежень.

Крок 5: побудова моделі процесу розробки на основі виділених вихідних даних.

Крок 6: аналіз моделі та, у разі необхідності, уточнення обмежень.

У загальному вигляді алгоритм вирішення задачі адаптації гнучких методологій з використанням методів інтелектуального аналізу даних представлений на рисунку 3.5.

Представимо процес у вигляді графу P, вершини якого відображають стан S системи, що моделюється, в різні моменти часу. Ребра графу R відображають дії системи. Функціонування процесу починається з початкового стану і моделюється переходами між вершинами по ребрах S_0:

$$r_j \in R, \mathrm{j} = \overline{1, J}. \tag{3.1}$$

При цьому з кожним таким переходом r пов'язані конкретні дії a_r з множини A допустимих дій процесу:

$$R \subseteq S \times A \times S. \tag{3.2}$$

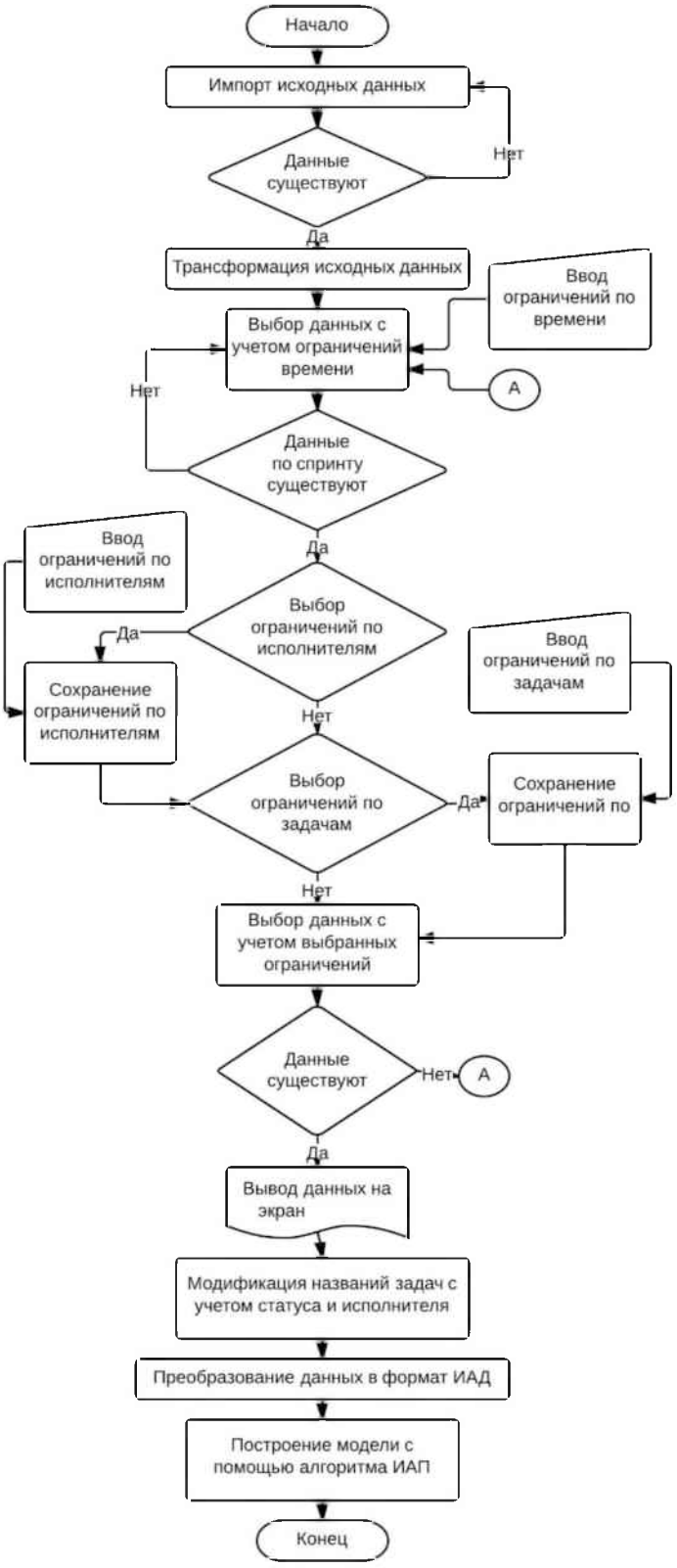

Рисунок 3.5 – алгоритм розв’язання задачі адаптації гнучких методологій до процесів розробки ПЗ з використанням ІАП

Таким чином, процес описується графом, вершини якого відображають стан модельованої динамічної системи, а дуги – переходи між цими станами:

$$P = (S, s_0, R, \mathrm{A}, \mathrm{O}) \tag{3.3}$$

Функціонування процесу є послідовністю переходів між його станами, причому кожен з таких переходів пов'язаний з виконанням відповідних дій:

$$\begin{aligned} &s_0 \xrightarrow{a_0} \ \ldots\ s_i \xrightarrow{a_i} s_{i+1} \xrightarrow{a_{i+1}} \ \ldots\ s_I, \\ &s_i \in S, a_i \in A, i = \overline{0, I}, \end{aligned} \tag{3.4}$$

Спочатку процес знаходиться в стані s_0, а в подальшому – на кожному i-тому кроці переходить у стан s_i. Наступний перехід здійснюється наступним чином: процес в i-тому стані обирає дугу з дією a_i, яку може бути виконано в поточний момент часу. Після виконання дії a_i процес переходить в наступний стан s_{i+1}. Процес продовжується до досягнення одного з кінцевих станів.

Виконання одного з можливих в поточному стані дій процесу залежить від вхідного стану процесу та набору виконаних дій. Кінцевим станом є такий стан, з якого відсутні дуги в інші стани процесу.

Реалізація кожного процесу (слід, траса процесу) являє собою послідовність дій, які переводять процес з початкового стану в кінцеве. У загальному випадку кожен процес має скінчену множину трас. Слід процесу відбивається в розглянутому вище файлі журналу.

Для кожного процесу P можуть бути задані обмеження з урахуванням атрибутів операцій. Обмеження за об'єктами визначаються наступним чином: нехай P – це процес, а O^* – підмножина об’єктів, над якими виконуються операції A цього процесу. Тоді обмеженням процесу P по множені об'єктів O^* є граф P^*, який отримано з графу P шляхом видалення тих операцій, які пов'язані з обробкою об'єктів з підмножини O^*.

3.5 Дослідження ефективності застосування методів Process mining

В гнучкою методології Scrum беклог спринту відображає список завдань, які повинні бути виконані під час поточної ітерації (спринту). Такі списки в основному представляють у вигляді карток на стіні – так званої Scrum-дошки або дошки завдань [12].

Методологія дозволяє відстежувати прогрес в ході роботи, використовуючи список завдань, представлених на дошці. Scrum дозволяє використовувати будь-які назви для станів завдань, але можна виділити стандартний набір станів [20]:

– To do – завдання знаходиться в черзі на виконання;

– Blocked – завдання заблоковано з будь-яких причин;

– In development – завдання виконується;

– Ready for testing – завдання вже виконано та готове до тестування;

– In testing – завдання знаходиться на стадії тестування;

– Done – завдання протестоване та повністю завершено.

Для використання Scrum-дошки командою вона ділиться на кілька колонок, кожна колонка відповідає статусу завдання. Використовуються паперові картки, на яких пишуться всі історії та завдання. На момент планування ітерації всі картки розташовуються в колонці «To do» відповідно до пріоритетів. Кожен день, коли член команди бере нову задачу, відповідна картка із завданням пересувається у колонку «In progress». На наступному щоденному мітингу кожен, хто закінчив роботу над деяким завданням, переміщує відповідну картку в колонку «Ready for Testing», і тестувальник може взяти цю задачу для тестування.

Така візуалізація процесу являє собою потужний інструмент, який може допомогти побудувати хороший Scrum-процес. Можна виділити наступні переваги:

– все наочно – дошка висить на стіні в кімнаті команди або в тому місці, де щодня проходить щоденний мітинг;

– відразу видно прогрес – окинувши поглядом дошку, можна відразу зрозуміти, скільки приблизно ще залишилося роботи;

– відразу видно перешкоди – наприклад, коли завдання беруться знизу і пріоритети ігноруються, або, наприклад, коли завдання висить в колонці «In progress» більше, ніж відведено їй часу, і багато інших «індикаторів», які допоможуть команді своєчасно зреагувати.

Сучасні системи контролю за процесом розробки ПЗ надають віртуальні Scrum-дошки, які представляють собою веб-сторінку з віртуалізованою Scrum-дошкою. Всі учасники процесу мають інтерактивний доступ до сторінки і можуть змінювати стани своїх завдань (рисунок 3.6).

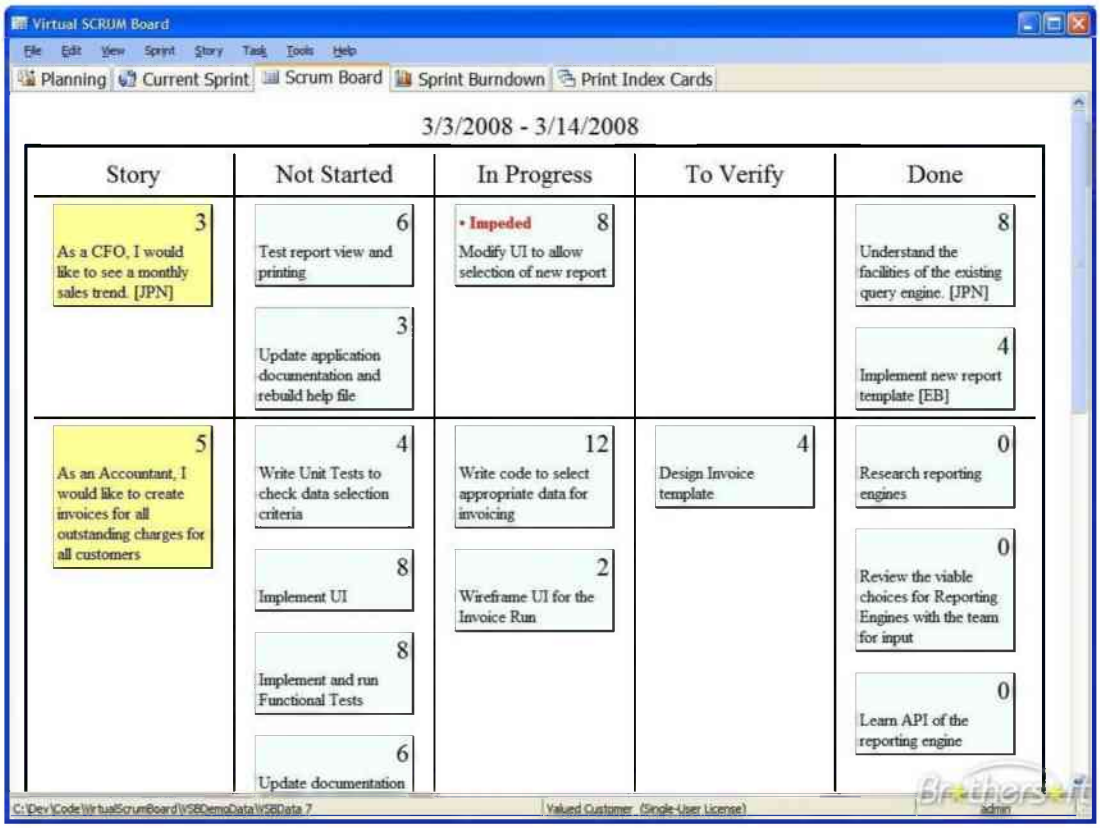

Рисунок 3.6 – Віртуальна Scrum-дошка

Основна відмінність Scrum-дошки від Kanban-дошки є те, що Scrum не забороняє команді поставити всі завдання в колонку «In progress» одночасно.

Проте, певне обмеження все ж існує, оскільки сама ітерація має певні обмеження за кількістю завдань.

Основний недолік такого способу візуалізації процесу є те, що Scrum-дошка описує як поточний стан прогресу, не враховуючи динаміку прогресу.

Використання методів Process mining дозволяє виявити послідовність виконання завдань та визначити динаміку прогресу в цілому. Такий підхід дозволяє виявити причинно-наслідкові зв'язки між переходами між станами завдань. В логах відображається реальне виконання процесу розробки програмного продукту. Застосування до логів методів Process mining дозволяє автоматично побудувати модель процесу. Побудовані таким чином моделі будуть відображати динаміку реального процесу розробки і доступні для сприйняття та аналізу.

4 ПРАКТИЧНЕ ВИКОРИСТАННЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

4.1 Інструментальні засоби інтелектуального аналізу проектів

На підставі проведеного аналізу [25] можна зробити висновок, що одним з найбільш поширених програмних продуктів для інтелектуального аналізу процесів є програмний продукт під назвою ProM. Цей програмний засіб реалізований на Java, тому він є незалежним від операційної системи. ProM має клієнт-серверну архітектуру, яка дозволяє розширювати функціонал програми за рахунок реалізації різних доповнень до програми (плагінів) [26]. На теперішній час існує понад 150 плагінів, які реалізують різні алгоритми і методи ІАП.

На рисунку 4.1 зображено початок роботи з додатком ProM.

Рисунок 4.1 – Вибір вихідних даних

Основними методами ІАП, які підтримуються в програмі, є методи control-flow (альфа-алгоритм та генетичний аналіз), методи аналізу організаційної перспективи (соціальні мережі), методи аналізу слабко-структурованих процесів (Фаззі-моделі) (рисунок 4.2).

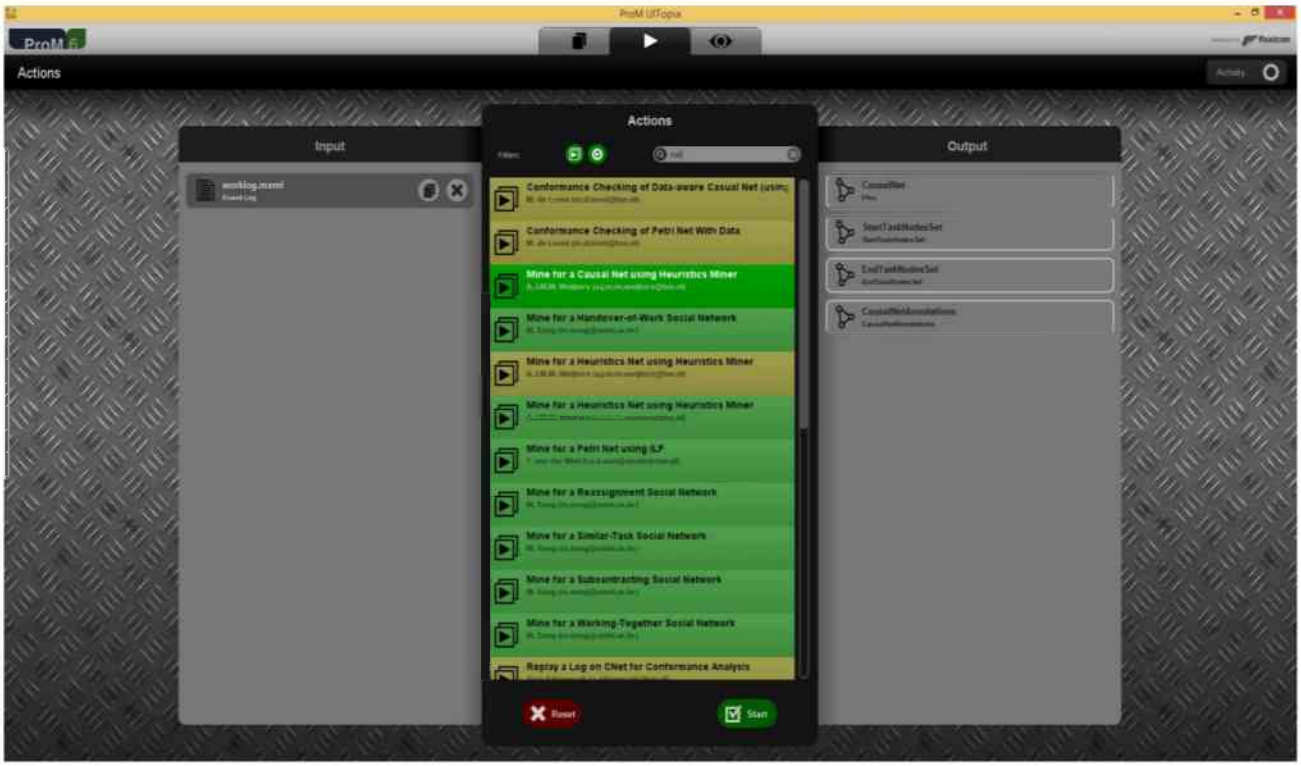

Рисунок 4.2 – Вибір методу інтелектуального аналізу даних

Разом з перевагами ProM містить ряд недоліків, серед яких:

– низька швидкість аналізу;

– під час оброки великих обсягів даних є вірогідність відмов, особливо у разі недостатніх ресурсів.

4.2 Аналіз журналів за допомогою програмного продукту ProM

У якості моделі бізнес-процесів оберемо процес розробки веб-магазину та проаналізуємо файли логів Scrum-дошки. Оскільки формати файлів журналів не відповідають MXML формату, за допомогою інструменту ProMImport

здійснимо їх перетворення в один файл. Цей файл містити результати журналювання одного процесу (розробка веб-магазину) (див. Додаток А до кваліфікаційної роботи) з п'ятьма станами, фрагмент якого наведено на рисунку 4.3.

```
<AuditTrailEntry>
<WorkflowModelElement>/trunk/www/index.html</WorkflowModelElement>
<EventType>complete</EventType>
<Timestamp>2017-11-09T13:07:41.000+01:00</Timestamp>
<Originator>tfmorris</Originator>
</AuditTrailEntry>
</ProcessInstance>
<ProcessInstance id="Case1.1">
<AuditTrailEntry>
<WorkflowModelElement>/trunk/src/org/argouml/language/cpp
/ui/SettingsTabCpp.java</WorkflowModelElement>
<EventType>complete</EventType>
<Timestamp>2017-11-10T10:03:23.000+01:00</Timestamp>
<Originator>mvw</Originator>
</AuditTrailEntry>
```

Рисунок 4.3 – фрагмент журналу Scrum-проекту до перетворення

Цей документ містить шляхи, де зберігаються частини проекту, та різні коментарі, що супроводжують проект. З метою виключення з аналізу локальних особливостей проекту, виконаємо заміну специфічних шляхів на абстракції: у наведеному фрагменті файли проекту з розширенням ".java" зберігаються у каталозі "/src/org/argouml/language/cpp/ui/", який ми замінимо на узагальнений шлях "SRC", а всі файли типу "readme.*" замінимо на "README", аналогічно перетворимо: "/tests/" – в "TESTS", "/www/" – в "WWW", "build.bat" – в "BUILDER" та всі файли, що починаються з "." – в файли "CONFIG". Всі інші

записи, що не відносяться до процесу розробки ПЗ, ми ігноруємо. Таким чином, ми отримали абстрактний лог-файл, що відповідає нашому бізнес-процесу.

Завантаживши цей файл в ProM виконаємо генерацію моделі у вигляді мережі Петрі (рисунок 4.4). На рисунку зображено спрощену мережу Петрі без циклів, яка була отримана із застосуванням стратегії модифікації моделі "Kill Loops" для перехідної системи та синтезуючи з нього мережу Петрі.

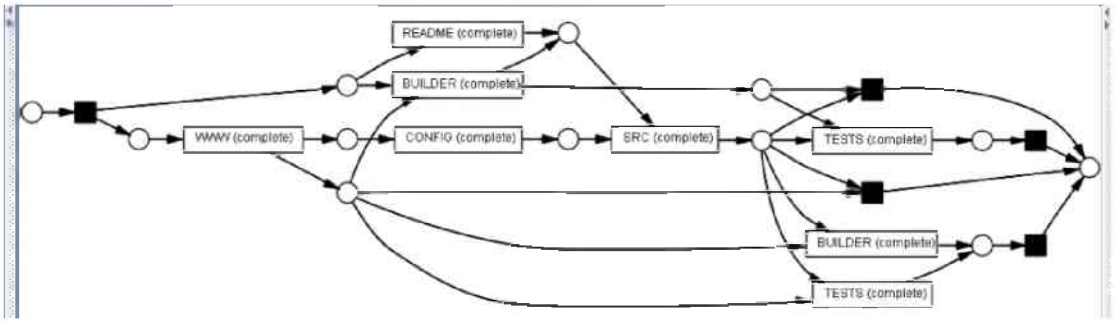

Рисунок 4.4 – Результат побудови мережі Петрі в ProM

На підставі отриманих результатів можна зробити висновок, що отримана мережа Петрі зосереджується на початкових подіях, тобто коли почалася розробка коду. Члени Scrum-команди та Scrum Master використовували на початку роботи над веб-сайтом редагування файлів readme і builders, після чого вони пишуть код, а потім перевіряють його, іноді builders змінюється після написання коду. Тепер, оскільки ми маємо модель, її можна розширити для роботи з часом та для представлення статистичних даних про тривалість завдань (рисунок 4.5).

Модель мережі Петрі процесу розробки тепер може бути використана для розширеного аналізу в рамках системи ProM. На рисунку 4.5 показаний результат аналізу продуктивності на основі вивчення моделі та журналу. Стани, тобто місця, були забарвлені відповідно до часу, який витрачається на них під час виконання процесу. Також декілька дуг, що походять з одного й того ж

місця (тобто варіантів), були анотовані до їх відповідної ймовірності (тобто часткою тих випадків, для яких цей шлях було обрано).

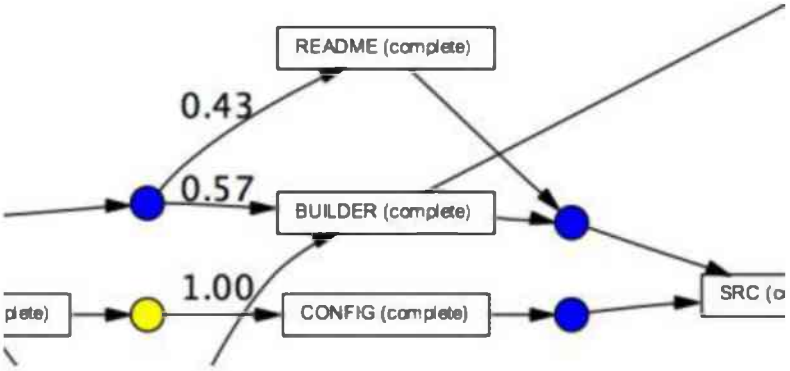

Рисунок 4.5 – Аналіз даних про витрати часу

Крім того, аналіз відповідності може бути виконаний з використанням мережі Петрі та відповідного журналу. На рисунку 4.6 наведено аналіз охоплення шляхів перевірки відповідності. Всі дії, виконані в конкретному випадку (або наборі випадків), оформлені напівжирним, а дуги містять позначки про частоту, з якою вони використовувались. У цьому прикладі можна побачити, що «README» жодного разу не використовувалась для «CPP», тобто команда з підтримки мови C ++ не створювала файл README і до роботи не приступала.

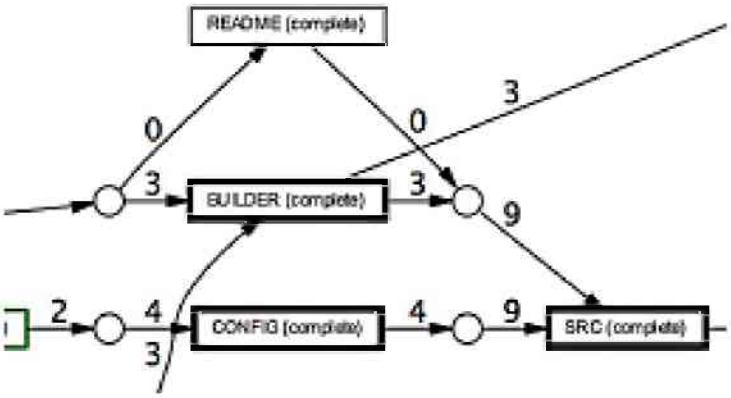

Рисунок 4.6 – Аналіз перевірки відповідності

Одним з обмежень у проекті розробки програмного забезпечення може бути те, що розробники, які працюють над вихідним кодом, не повинні виконувати тестування. На рисунку 4.7 показано результат перевірки журналу проекту. У випадку підтримки мови Java, фрагмент якої наведено на рисунку 4.7, розробник «euluis» подав як вихідний код, так і тести, тим самим порушуючи це обмеження.

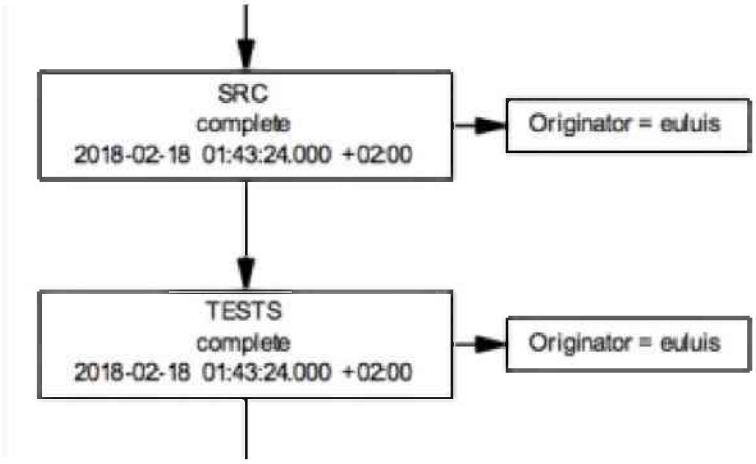

Рисунок 4.7 – Аналіз Scrum-проекту

З метою визначення соціальних зв'язків між учасниками команди проекту, що може бути доцільним для оптимізації складу команди розробників, для аналізу процесу розробки бажано використовувати повний журнал, тобто перед його абстрагуванням. Рисунок 4.8 ілюструє передачу робіт між розробниками. Аналіз цієї моделі свідчить про той факт, що деякі з розробників беруть участь в роботі над проектом лише на конкретних етапах проекту: наприклад, «bobtarling» працює тільки на прикінці проекту, тоді як інші члени команди (наприклад, «tfmorris») виконували різні завдання протягом усього процесу розробки ПЗ.

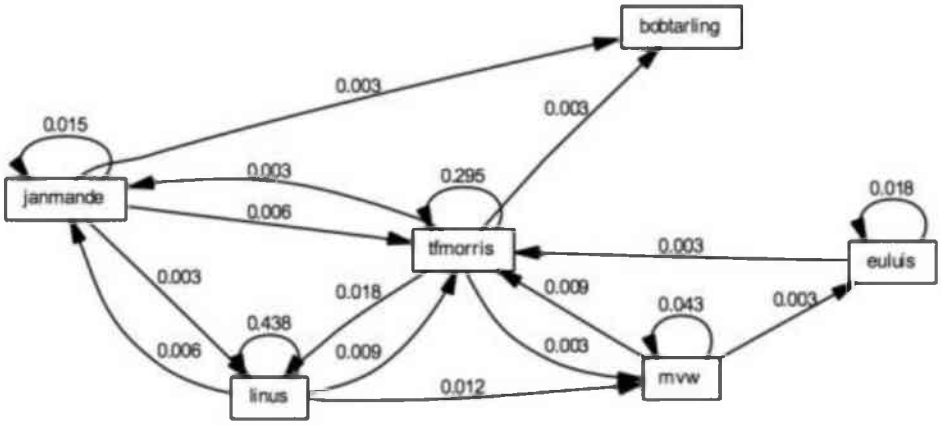

Рисунок 4.8 – Соціальна активність учасників проекту

Таким чином на підставі проведеного аналізу з використанням інструментального засобу ProM можна оцінити ефективність використання методологій швидкої розробки ПЗ. З метою оцінювання було використано аналіз журналів про виконання програмних процесів під час розробки веб-додатку, оскільки він є типовим для розробки невеликих проектів. Проведене моделювання доводить, що ми можемо отримати досить адекватну моделі бізнес-процесів, що супроводжують розробку програмних систем. Крім того, під час дослідження було продемонстровано варіант використання ProM для аналізу та перевірки деяких особливостей управління процесом розробки ПЗ.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи, відповідно до мети і завдань, були отримані наступні основні наукові результати:

– виконано дослідження особливостей застосування гнучких методологій управління;

– проаналізовано сучасні засоби управління для автоматизації проектів;

– описано сучасні методи управління проектами;

– здійснено постановку задачі дослідження;

– виконано розробку алгоритму вибору гнучких методів управління проектами в ІТ-компанії;

– запропоновано шляхи покращення процесу розробки проектів із застосуванням гнучкої методології Agile.

В кваліфікаційній роботі описано технічні рішення і організаційно-технічні заходи, які забезпечують виконання задачі проектування ПЗ. Проаналізовано визначення термінів «проект» і «ІТ-проект» для виявлення відмінних характеристик, було з'ясовано, що управління ІТ-проектом набагато відрізняється від управління проектом, не пов'язаним з інформаційними технологіями. Було розглянуто поняття «життєвий цикл ІТ-проекту» та проаналізовані переваги і недоліки гнучких методологій. Проведено аналіз специфіки предметної області дослідження на предмет застосування тієї чи іншої моделі життєвого циклу при управлінні ІТ-проектом. Як наслідок, була висунута гіпотеза про ефективне застосуванні «гнучкою» методології управління ІТ-проектом.

В рамках виконання кваліфікаційної роботи були проаналізовані найтиповіші особливості проектів та проведено їх порівняльний аналіз. В результаті були виділені основні типи проектів залежно від робіт, які виконує команда проекту. Таким чином, вдалося поєднати інструменти Agile і особливості реалізації програмних проектів в ІТ-компаніях. Розроблений алгоритм призначений для ІТ-компаній, які бажають використовувати гнучке

управління проектами, і призначений для підбору інструменту гнучкого управління залежно від умов, в яких працює команда. В алгоритмі враховано типи проектів, команд і самих замовників.

Результати, що були оримані під написання кваліфікаційної роботи, були висвітлені автором під час обговорення в рамках 22-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті», що проводився у Харківському національному університеті радіоелектроніки З 17 по 19 квітня 2018 р [27].

Подальший розвиток теми дослідження стосовно доопрацювання алгоритму вибору інструментарію за Agile-методологією передбачає розробку хмарного застосування, яке дозволить не лише обирати ту чи іншу методологію, а й рекомендувати більш популярну для розв’язання конкретних проектних рішень на підставі статистики їх використання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мартин П. Быстрая разработка программ. Принципы, примеры, практика. Tennessy, 2011.264 с.

2. Bosch J., Bosch-Sijtsema P. M. Introducing agile customer-centered development in a legacy software product line. Software: Practice and Experience, 2011. P. 871–882.

3. Мартин П., Тейт К. Управління проектами. Київ: Магістр, 2006. 224 с.

4. Расмуссон Дж. Гибкое управление IT-проектами. Руководство для настоящих самураев. Вильямс, 2009. 312 с.

5. Ткачук М.В., Гамзаев Р.О. Модель та інформаційна технологія побудови адаптивної матриці трасування вимог в гнучких процесах розробки програмного забезпечення. *Вісник НТУ «ХПІ»*. Харків. № 2 (976). 2013. С. 49–60.

6. Тверезовська Н. Т., Кишенко В. А. Управління проектами: стан і перспективи розвитку: підруч. [для студ. вищ. навч. закл.]. Київ: Юрайт, 2014. 161 с.

7. Project Management Methodology: Definition, Types, Examples. URL : http://www.mymanagementguide.com (дата звернення 12.10.2021).

8. Пушкарьов О. М. Гнучка методологія розробки проектів: підруч. [для студ. вищ. навч. закл.]. Київ: Бізнес-школа, 2014. 84 с.

9. PMBOK Guide. 5th ed. Newton Square, Pennsylvania, USA: Project Management Institute, 2013.

10. Process Mining: Discovery, Conformance and Enhancement of Business Processes. *W.M.P. van derAalst*. N.–Y.: Springer Verlag, 2011. 370.

11. SWEBOK. Guide to Software Engineering Base of Knowledge (SWEBOK). IEEE Computer Society, 2004. URL: http://www.swebok.org/ (дата звернення: 16.10.2021).

12. Кейтон А. Теорія пізнання «Scrum». Харків: Фенікс, 2016. 340 с.

13. ISO/IEC 25010:2011. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software

quality models. URL: https://www.iso.org/standard/35733.html (дата звернення: 16.10.2021).

14. ISO/IEC 14143-6:2012. Information technology. Software measurement. Functional size measurement. Part 6: Guide for use of ISO/IEC 14143 series and related International Standards. URL: https://www.iso.org/standard/60176.html (дата звернення: 16.05.2018).

15. COCOMO II Model Definition Manual – USC CSSE. URL: http://csse.usc.edu/TECHRPTS/2000/usccse2000-500/usccse2000-500.doc (дата звернення: 16.10.2021).

16. Process Mining. URL: http://www.processmining.org. (дата звернення: 16.11.2021).

17. Process Mining: Beyond Business Intelligence. URL: http://www.processmining.org/_media/presentations/processminingtutorialesscass-2009.pdf (дата звернення: 16.10.2021).

18. C. Effort Agile Software Development. LA, 2010. 256 с.

19. Cohn M. - Succeeding with Agile. Tennessy, 2012. 239 с.

20. Визначення «Scrum»: Правила гри.: [авт. тексту Іменко М.]. К.: Наукова книга, 2013. 119 с.

21. How & Why to Use the Kanban Methodology for Software Development URL: https://www.sitepoint.com (дата звернення 09.10.2021).

22. Aals van der W. Workow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*. 2004. № 16 (9). P. 1128–1142.

23. Cohn M. Agile checklist. Tennessy, 2011. 123 с.

24. Alexey Mitsyuk, Anna Kalenkova, Sergey A. Shershakov, van der Aalst W. Using process mining for the analysis of an e-trade system: A case study. *Бизнес-информатика*. 2014. Vol. 29. No. 3. P. 15–27.

25. Top 13 Project Management Methodologies and Style. URL: http://blog.azendoo.com (дата звернення: 04.10.2021).

26. ProM 6 Tutorial. URL: http://www.promtools.org/prom6/downloads/prom-6.0-tutorial.pdf (дата звернення: 24.10.2021).

27. Билоенко О.В. Исследование методов интеллектуального анализа процессов в программных задачах разработки проектов с использованием Agile-методологии // 22-й Міжнародний молодіжний форум «Радіоелектроніка і молодь в XXI столітті», Зб. матеріалів форуму, Ч.1. Харків: ХНУРЕ, 2018. С.143–144.