Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Факультет інформатики
Кафедра інформатики

Магістерська робота

Освітній ступінь: магістр

На тему: «АГЕНТНО-БАЗОВАНИЙ ПІДХІД ДО МОДЕЛЮВАННЯ КОЛЕКТИВНОЇ ДІЯЛЬНОСТІ»

Виконала: студентка 2 року навчання
Спеціальності
122 Комп’ютерні науки
Радзієвська Олександра Володимирівна
Керівник
Гороховський С.С.
кандидат фіз.-мат. наук

Рецензент М. М. Глибовець
Магістерська робота захищена
з оцінкою_____
Секретар ЕК С.А. Мелещенко
«___» _____2022

Київ 2022

Зміст

Список використаних зображень

Анотація

Дана робота присвячена використанню агентно-базованого підходу для моделювання колективної діяльності, а саме для моделювання роботи команди розробників над проєктом зі сфери IT. Для характеристики агентів використовуються їх професійні навички, мотивація, готовність просити про допомогу та допомагати іншим в середині команди. Результатом роботи є модель, що дозволяє проводити експерименти з метою дослідження процесів, що відбуваються в середині команди.

Вступ

Життя людини, як соціальної істоти, неможливе поза суспільством. Все наше життя наповнене величезною кількістю соціальних взаємодій, що керуються суспільними нормами та правилами. Ми постійно знаходимось в контакті з іншими людьми, працюємо над спільними проблемами та виконуємо відведену в суспільстві роль. У процесі таких взаємодій виникає безліч феноменів, вивчення та розуміння яких важливе для підтримки порядку та створення максимально комфортних умов для всіх учасників соціуму.

У даній роботі на прикладі команди IT розробників буде розглядатись соціальне явище командної діяльності.

В умовах зростаючої конкуренції та обмеженості ресурсів, команди, які працюють спільно, можуть отримати більше можливостей для досягнення цілей [1]. Це особливо актуально в сьогоднішній час, коли інтелектуальних можливостей окремого індивіда недостатньо для досягнення значних проривів в більшості сфер. У такому випадку колективна робота є й природнім способом подолання обмежень людського інтелекту.

Колективна робота починається з формування команди. Задача пошуку найкращого кандидата досить нетривіальна, адже кандидат має володіти не лише потрібним рівнем hard skills (знань безпосередньо з області, в якій йому доведеться працювати), але й soft skills (наскільки людина поділяє цінності компанії, здатна працювати в колективі тощо). Ще важчою задача стає, якщо спробувати оцінити кандидата не просто з погляду загальних вимог компанії, а того, як саме ця людина впишеться в конкретну команду, враховуючи характери членів команди, когнітивні стилі, вмотивованість, рівень відповідальності, здатність навчатись тощо.

Тим не менш, такий глибший погляд міг би привести до значно кращих результатів, як для компанії, яка наймає співробітників, так і для співробітників, які отримують краще середовище для роботи.

Існує багато методів та інструментів, що дозволяють класифікувати чи охарактеризувати числовим значенням особливості окремої людини (тест IQ, тести на когнітивні здібності, тип темпераменту тощо). Та мало досліджено те, як люди з різними особливостями характеру взаємодіють між собою, особливо в професійному плані. Стандартні соціологічні методи, які могли б бути використані для такої задачі, затратні за часом та ресурсами, адже спостереження можуть тривати роки. До того ж, щоб отримати результати, які були б релевантними не лише для однієї конкретної компанії, потрібно проводити дослідження на величезній кількості учасників з різних контекстів [2]. Альтернативним методом, що дозволяє з мінімальними витратами провести початкові експерименти, відкинути нежиттєздатні варіанти, сформулювати гіпотези, тим самим значно спростивши та пришвидшивши подальші дослідження, є моделювання з використанням агентно-базованого підходу.

Отже, метою роботи є за допомогою агентно-базованого підходу розробити модель, що імітує роботу команди розробників над проєктом. Створити програму, що дозволяє експериментувати з характеристиками членів команди та задачі для проведення власних досліджень з використанням моделі.

Текстова частина роботи складається з трьох розділів.

У першому описується поняття агента, агентно-базованого моделювання, його переваг та доцільності використання в даній задачі; проводиться огляд існуючих досліджень на схожу тему.

Другий розділ оглядає предметну область - поняття проєкта, проєктного менеджменту, методологій розробки. Також виконується

спроба виділити та описати ті людські риси, що впливають на її роботу в команді.

Третій розділ присвячено опису моделі реалізованої в рамках роботи. Описано архітектуру проєкту, особливості побудови агентів та середовища, логіку їх взаємодії. Окремо оглядаються технології, які були використані при розробці.

РОЗДІЛ 1. АГЕНТНО_БАЗОВАНЕ МОДЕЛЮВАННЯ

1.1 Поняття агента

У загальній лексиці “агент” означає особу, яка діє від імені іншої особи або групи. Для програмованого агента не існує єдиного визначення, різні дослідники пропонують власні трактування, і навколо цього постійно точаться дебати.

Ерік Бонабо вважає агентом будь-яку автономну сутність, що здатна приймати рішення на основі певних правил [3].

Вулдрідж та Дженнінгс ставлять вже більше вимог до поняття агенту і пропонують у своїй роботі [4] два визначення. М’якше - визначає агента, як апаратний чи, частіше, програмований комп’ютерний елемент, що відповідає наступним вимогам: автономність, наявність соціальних навичок (вміння комунікувати), реактивність, проактивність. Більш строге визначення, окрім перелічених рис, вимагає від агентів ще наявності таких притаманних людям рис, як вірування, сподівання, зобов’язання та наміри.

У пізнішій своїй праці [5] Вулдрідж, посилаючись на ці визначення, узагальнює їх до більш короткого “Агент - це комп'ютерна система, яка розташована в певному середовищі і здатна автономно діяти в цьому середовищі для досягнення поставлених цілей”. При цьому передбачається що агент має певні сенсори для отримання даних з навколишнього середовища та актуатори, для виконання певних дій в середовищі. Загально кажучи під такий опис може потрапляти багато досить простих систем, як наприклад кондиціонер, що починає працювати на нагрівання або охолодження, коли температура в кімнаті досягає певного значення.

Йоав Шохам визначає агента як сутність, стан якої складається з ментальних компонентів, таких як переконання, можливості, вибір та

зобов’язання [6]. Тобто також акцентує увагу саме на наявності “людських” рис.

Інші дослідники включають у свої визначення інші додаткові характеристики, або вважають що якісь з озвучених вище не є обов’язковими. Ці суперечки тривають ще й тому, що для різних предметних областей, вимоги (і відповідно архітектура) до агента можуть бути різними. Наприклад якщо у нас є агент, що має пересуватись поверхнею Марсу і збирати зразки порід, нам необхідно щоб у агента була функція навчання, і він змінював свої маршрути відповідно до набутих знань про особливості рельєфу. Натомість, при використанні агента що оцінює якість повітря, нам в ніякому випадку не можна допустити щоб цей агент якось змінив свої початкові налаштування і реакцію на подразнення, адже він має показувати точні дані, як це було передбачено початковими налаштуваннями [5].

Група агентів, що взаємодіють між собою в певному середовищі для досягнення спільних або суперечливих цілей утворюють багатоагентну систему (multi-agent system, MAS). MAS є основною сферою досліджень сучасного штучного інтелекту [7].

Ефективність використання агентного підходу підвищується для алгоритмів, де від агентів вимагається поводити себе на кшталт того, як поводять себе люди, а не діяти оптимально до задачі.

Мультиагентні системи широко використовуються для розв’язання прикладних задач в багатьох сферах, зокрема електроенергетиці [8], економіці [9, 10], плануванні дій на випадок природних та техногенних катастроф [11, 12], захисті критичної інфраструктури [13, 14, 15], архітектурі [16], урбаністиці [17, 18], розробці ігор [19], транспортних задачах та аналізу трафіку [20, 21], географії [22], картографії [23], електронній комерції [24] та інших.

1.2 Агентно-базоване моделювання

Моделювання - один з теоретичних методів наукового дослідження, що являє собою вивчення якогось об’єкту чи явища за допомогою моделі - спрощеного представлення оригіналу, зі збереженням ключових рис досліджуваного об’єкта. Цей метод особливо актуальний, коли проведення емпіричних досліджень - спостережень та експериментів над реальним об’єктом є неможливим або недоцільним. Наприклад, спостереження за якимось явищем протягом багатьох років, моделювання поведінки великих соціальних груп тощо.

Моделювання з допомогою мультиагентних систем (Agent-based modeling, ABM) - відносно новий спосіб моделювання, що використовує агенти - програмовані компоненти, що здатні автономно діяти в середовищі та взаємодіяти один з одним. Варто розрізняти ABM від звичайної багатоагентної системи. Мета агентного моделювання - з допомогою простих правил знайти пояснення складних феноменів колективної поведінки. Багатоагентна система зазвичай покликана розв’язати якусь конкретну практичну або технічну проблему [25].

Цінність моделювання з допомогою агентів - в значній економії ресурсів. За допомогою відносно мінімальних затрат можна відкинути нежиттєздатні варіанти і в емпіричних дослідженнях сфокусуватись лише на тих опціях, що показали перспективні результати при моделюванні.

ABM не просто технологія, а спосіб мислення про проблему. Осягнути та описати певний процес складно, адже це абстрактне поняття. Натомість розглянути окремі компоненти, що породжують цей процес, та описати їх активності зазвичай досить легко. Наприклад, природніше описати, як покупці пересуваються в супермаркеті, ніж придумати рівняння, які регулюють динаміку щільності покупців [3]. Крім того, такий підхід інтуїтивно зрозумілий, адже дослідникам легко асоціювати себе з

агентами, та водночас дозволяє розуміти глибинні процеси системи та досліджувати складні поняття як явище емерджентності (“emergent phenomena”).

Поняття емерджентності означає, що за рахунок зв’язків між складовими, система набуває властивостей, не властивих окремим її компонентам. Тобто явище, що виникає за рахунок взаємодії агентів, може мати властивості, яких не мають агенти поодинці [3]. Наприклад, натовп людей може рухатись в протилежну сторону, ніж окремі люди, які знаходяться в цьому натовпі і утворюють його. Через свою особливість emergent phenomena важко передбачити, та ABM дозволяє досліджувати його досить природним способом.

У всіх методів моделювання є спільна особливість, яка полягає в тому, що модель повинна мати конкретну задачу. Найчастіше модель загального призначення не виконуватиме своїх функцій [3]. До того ж у моделі зазвичай неможливо відтворити всі риси реального об’єкта, тому виділяються лише ті відомі знання про об’єкт, які важливі для конкретного дослідження. Наприклад при моделювання командної роботи групи людей, важливим буде задати інформацію про соціальні здібності та рівень експертності (досвідченості) людини в конкретному завданні, але інформація про хобі людини та рівень її володіння латиною навряд знадобляться. Хоча в реальному світі незначні деталі й можуть мати вплив, в симуляції вони створюватимуть зайвий шум. До того ж, в першу чергу, модель покликана дослідити загальні тенденції.

При розробці моделі дослідник наділяє агентів рисами та поведінкою, які він вважає за потрібне, спираючись на певні дослідження та свій власний досвід. Зрозуміло, що такі рішення є досить суб’єктивними, а отже ми не можемо бути впевненими, що модель повністю відображає реальні процеси. Тому результат досліджень з використанням агентно-базованого підходу – це, зазвичай, не підтверджені теорії, а швидше гіпотези - припущення про

явище, яке мають бути підтверджені, або спростовані за допомогою соціологічних досліджень з реальними групами людей.

Оскільки ключовою ідеєю моделювання з використанням агентів є генерування складної поведінки простими правилами, при роботі важливо дотримуватись принципу K.I.S.S. (“Keep it Simple, Stupid”).

1.3 Використання агентно-базованого моделювання для дослідження соціальних явищ

Соціальне життя наповнене різними формами взаємодій між непов'язаними індивідами. Часто інтереси окремої особи розходяться з тим, що є кращим для суспільства, такі ситуації регулюються соціальними нормами та інституціями.

Наука, що займається дослідженням різних процесів в суспільстві називається соціологією. Методологія соціологічних досліджень включає в себе різні методи збору інформації, зокрема спостереження, анкетування, проведення інтерв’ю, метод експерименту, експертної оцінки тощо, для подальшого аналізу та дослідження. Жоден з цих методів не є універсальним, має чітко окреслені межі пізнавальної можливості, а отже і сферу застосувань. В окремих випадках, стандартні соціологічні інструменти не здатні покрити всіх потреб дослідників, або є малоефективними. Наприклад при звичайному спостереженні за групою людей важко відокремити який саме показник мав найбільший вплив і спричинив певне явище. У такому разі часто використовують моделювання, зокрема моделювання з використанням агентів.

Вперше агентно-базованого моделювання було використане як інструмент для дослідження соціологічних явищ у 60-х роках минулого століття. Піонерами цієї області вважаються соціологи Джеймс С. Коулман

і Реймонд Будон. У 1971 році було опубліковано статті Джеймса М. Сакоди та Томаса С. Шеллінга про динаміку сегрегації. Та справді велика кількість досліджень на цю тему та готових інструментів з'явилась лише з розвитком комп'ютерних потужностей у 1990-х. Тогочасні ABM платформи пропонували функціонал для моделювання таких соціальних явищ як виникнення політичних коаліцій, сегрегації, кооперування, формування колективної думки тощо. Причому функціонал був достатньо зрозумілий, так що скористатись цим інструментом могли люди без особливих знань в області комп'ютерних наук [26].

Сьогодні агентно-базоване моделювання використовується для соціальних досліджень різних напрямків, зокрема й для дослідження колективної діяльності. У таких дослідженнях зазвичай розглядається те, як риси членів команди впливають на їх здатність працювати разом, та які підходи до формування команд можуть гарантувати вищу продуктивність.

Наприклад, у статті [27] автори досліджують процес розв'язання задачі командою. Для цього вони намагаються моделювати процес розв'язання групою агентів Subarctic Survival Situation – задачі, в умові якої вимагається розташувати 15 предметів у порядку важливості для виживання в Арктичній пустелі. Зазвичай учасникам пропонується самостійно створити рейтинг цих предметів, а потім обговорити результати з командою, знайти і представити найкращий результат. У симуляції дослідники наділяють агентів наступними характеристиками: talkativeness (бажання представити свої варіанти та судження команді), agreeableness (готовність приймати альтернативні рішення інших членів команди) і critical thinking (здатність до критичного мислення).

У результаті роботи було отримано модель та проведено експерименти, з яких автори дійшли висновку, що для досягнення консенсусу більш важливим є готовність комунікувати, а не інтелектуальні здібності учасників. Причому високий рівень agreeableness пришвидшує

пошук єдиного рішення, а високий рівень talkativeness навпаки відтягує цей момент.

Автори іншої статті [28] працюють над вивченням того, як когнітивний стиль людини впливає на її роботу в команді, та люди з якими когнітивними стилями краще співпрацюють між собою. Для того, щоб охарактеризувати людину за її когнітивним стилем, вони використовують інструмент KAI (Kirton Adaption-Innovation Inventory). Цей метод дозволяє оцінити стиль розв'язання задач людиною по шкалі від високо адаптивного до високо інноваційного.

Цілями описаного дослідження є:

1) побудувати агентно-базовану модель, агенти якої імітують роботу людей з різними когнітивними стилями;

2) використовуючи модель, порівняти ефективність розв'язання задачі командою та її учасниками окремо;

3) оцінити як когнітивний стиль учасників впливає на спеціалізацію команди та комунікацію всередині неї.

У результаті автори презентують модель KABOOM - перший агентно-базований фреймворк, що дозволяє вивчати те наскільки якісно люди з різними когнітивними стилями можуть співпрацювати над розв'язанням проблем. Крім того дослідники діляться власними результатами експериментів на тему того, в командах якого розміру краще працювати високоадаптивним та високоінноваційним агентам.

Метою даної роботи є не дослідження впливу якоїсь конкретної риси характеру на результати команди, а симуляція загального процесу розробки. Схожим дослідженням займались автори статті [2]. Розроблений ними фреймворк вміє моделювати індивідуальну та колективну роботу над задачами, а також формальне та неформальне спілкування в середині команди. Алгоритм роботи фреймворку наступний. На вхід подається склад команди та опис проєкту (з точки зору робочих процесів), модель імітує два

тижні роботи цієї команди над цим проєктом і віддає на вихід інформацію про те які зміни відбулись в команді та проєкті за цей час (які задачі було виконано та хто над ними працював, хто що вивчив, хто з ким комунікував, у кого змінилась мотивація, до кого зменшилась чи збільшилась довіра тощо).

Також автори провели дослідження у якому протягом двох тижнів слідкували за роботою справжньої команди з 15 людей, щоб виміряти якими активностями працівники займаються протягом дня, з ким спілкуються та співпрацюють, їх мотивацію тощо. Дослідники використали ці дані для перевірки коректності роботи свого алгоритму.

Автори вказують, що створений фреймворк є лише першою версією моделі і вони працюють над збільшенням часу симуляції (наразі можливо лише симулювати одну двотижневу ітерацію) та кількості досліджуваних рис. Також вони хочуть провести більш ґрунтовні дослідження, адже наразі отримані в симуляції результати можна застосувати лише для однієї конкретної організації.

Висновки до розділу 1

Мультиагентні системи надають прості рішення для складних задач у багатьох сферах. Зокрема, agent based modeling - потужний інструмент, що дозволяє значно скоротити ресурсні, часові та грошові витрати на дослідження в соціологічній сфері. Найскладнішим завданням при моделюванні є знаходження відповідного рівня абстракції, який охоплює достатню кількість деталей, уникаючи при цьому отримання результатів, які неможливо інтерпретувати.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

У даній роботі за допомогою агентно-базованого підходу буде створено модель, що імітуватиме роботу команди, що складається зі спеціалістів різних спеціалізацій, над певним проектом в IT галузі. Очікується, що за допомогою такої моделі можна буде дослідити оптимальні розміри команди для певних задач, особливості взаємодії людей з різним типом темпераменту та інші глибинні соціальні процеси всередині команди, що має спростити керування проєктами.

Дві основні сутності, що фігурують в постановці задачі - команда і проєкт, над яким вона працює. Для початку розглянемо ці компоненти детальніше, щоб зрозуміти які риси можуть бути виділено як важливі для моделі, та як їх можна формалізувати. Після цього, розглянемо також поняття проєктного менеджменту та методологій, що використовуються при реалізації проєкту.

2.1 Поняття проєкту

Згідно з визначенням Інституту Управління Проєктами (PMI), проєкт - це тимчасові зусилля направлені на отримання унікального продукту, сервісу чи результату [31]. Важливою рисою проєкту обмеженість ресурсів, а також згадана у визначенні тимчасовість - у проєкту є чіткий початок і кінець, і він не повторюється після закінчення. Натомість, можна розпочати новий проєкт.

Командний проєкт можна розглядати як набір певних задач (процесів), завершення виконання яких означатиме завершення проєкту. Задачі розрізняються за спрямуванням, необхідними навичками для їх

виконання, обсягом тощо. Спробуємо класифікувати задачі для кращої формалізації поняття.

1. Сфера задачі. Дає загальну інформацію про те, що загалом являє собою задача та визначає, які спеціалісти мають кваліфікацію для її виконання. Наприклад створення макетів сайту - задача для дизайнера, а тестування готового функціоналу робота для QA спеціаліста.
2. Вміння, необхідні для виконання задачі. Кожна задача вимагає володіння певними знаннями - загальними уявленнями про область, технологіями та інструментами, що використовуються під час роботи. Наприклад, для отримання списку користувачів з бази даних потрібно знати принцип побудови запитів, синтаксис мови чи діалекту, який використовується та розуміти яку програму використати для виконання запиту.
3. Терміновість задачі, пріоритет. Наскільки задача важлива для користувача/клієнта. Зазвичай пріоритети оцінює менеджер (детільніше роль менеджера проєкту буде обговорено нижче), але він може коригуватися після обговорення з командою.
4. Рівень складності. Оцінка складності задачі перед її виконанням є важливим для розуміння часових меж, необхідних ресурсів, залежностей між задачами. Надати точну оцінку є складним, майже неможливим завданням, можна лише спробувати максимально її наблизити. Традиційно при оцінці використовувались часові виміри - дні, тижні, місяці. Та сьогодні більш популярним є використання сторіпоінтів (з англ. story points). Story point - це одиниця виміру для вираження оцінки загальних необхідних зусиль для реалізації завдання. Це значення відображає складність роботи, кількість роботи, рівень ризиків та невідомості. Використання сторіпоінтів переносить фокус з часу виконання на безпосередню складність

задачі та їх цінність [32]. При оцінці можна поставити наступні питання, що допоможуть зрозуміти рівень складності:

a. Чи задача поставлена достатньо чітко, чи зрозумілі всі деталі та який фінальний результат очікується?
b. Які підзадачі включає в себе задача та скільки часу орієнтовно займає виконання кожної з них?
c. Чи наявні всі ресурси, необхідні для його виконання?
d. Чи залежить задача від інших задач; чи є задачі, що мають бути завершені до початку виконання цієї задачі?
e. Хто ще має бути залучений до виконання цієї задачі, скільки часу займе виконання цієї частини задачі цією людиною?
f. Чи виконавці вже мають досвід виконання схожих завдань? Скільки часу вони зайняли?
g. Чи є фактори які можуть створювати затримки в роботі чи сповільнювати процес? [33]

5. Рівень творчості завдання. Відображає те, чи є завдання рутинним, чи вимагає нових підходів та ідей. Корелює з тим, люди з яким когнітивним стилем можуть краще себе проявити при виконанні цього завдання.
6. Порядок задач та залежності. Задачі проєкту не існують у вакуумі, так чи інакше більшість з них тісно пов’язані між собою - деякі потрібно виконувати паралельно, інші мають йти строго одна за одною. При плануванні необхідно зважати на це, щоб виконання задач не було заблоковане через те, що ще не завершені задачі, що мають їй передувати.

2.2 Команда проєкту

Зазвичай команда, що працює над проєктом складається зі спеціалістів різних спрямувань, кожен з яких має власний попередній досвід, набір особистих рис, сильних та слабких сторін. Усі ці якості впливають на те, які задачі, як якісно та як швидко вони можуть виконувати. Крім того, в умовах командної роботи, важливими є не лише знання і вміння цього розробника як індивіда, а й те як він вміє взаємодіяти з іншими членами команди, обговорювати, знаходити компроміси, підлаштовуватись тощо.

Виділимо основні риси членів команди, які відрізняють їх одне від одного та впливають на якість їхньої роботи.

1. Спеціалізація. Найчастіше при розробці програмного проекту виділяють back-end, front-end, QA, devops спеціалістів, а також бізнес-аналітиків ML інженерів, дизайнерів тощо. Загально кажучи, спеціалізація відповідає за наявність певних базових знань, достатніх для виконання певного класу задач. Деякі спеціалісти можуть виконувати декілька ролей всередині команди (наприклад брати як back-end, так і front-end задачі).
2. Вміння та попередній досвід. Кожен спеціаліст має досвід з виконання певних задач, а також набір технологій з якими він працював в минулому та/або має хороші теоретичні знання. Попередній досвід важливий, адже повторне виконання схожих задач зазвичай займає набагато менше часу ніж тих, з якими людина стикається вперше. Так само володіння технологіями зазвичай значно спрощує опанування схожих.
3. Загальний рівень здібностей/загальний досвід. Для того щоб якимось чином класифікувати рівень спеціаліста на практиці найчастіше використовують градацію intern, junior, middle, senior, tech lead тощо.

Така градація досить умовна, рівень спеціаліста в одній компанії може відрізнятись від іншої. Також можуть відрізнятись назви позицій, та кількість рівнів. Тим не менш, ця класифікація дає можливість загально розуміти глибину знань спеціаліста, а також дає нам інформацію про рівень завдань, які виконує ця людина всередині команди. Так завдання з проєктування архітектури зазвичай виконують спеціалісти високого рівня, такі як senior або tech lead. На противагу більш рутинні завдання дістаються спеціалістам початкових позицій.

4. Когнітивний стиль. Когнітивний стиль визначає індивідуальні відмінності людини в тому як вона сприймає інформацію, мислить, розв'язує проблеми, вчиться та взаємодіє з іншими людьми [29]. Когнітивні здібності людини можна оцінити за допомогою різних психометричних інструментів. Наприклад, для визначення того, як людина підходить до розв'язання задач використовують Kirton Adaption–Innovation Inventory (KAI). За результатами 32 питань цього тесту можна оцінити problem-solving стиль людини за шкалою від високо адаптивного до високо інноваційного. [28]
5. Мотивація. Мотивація відповідає за причини по яким людина вирішує виконувати певну дію, за те, скільки часу вона готова присвятити її виконанню та як багато зусиль прикласти для досягнення цілі [30]. На мотивацію можуть впливати рівень особистої зацікавленості людини в проєкті, грошова, репутаційна та інші винагороди за виконання (та навпаки - санкції, що будуть застосовані при невиконанні), наявність більш пріоритетних завдань тощо. Низький рівень вмотивованості може нівелювати високі показники знань та досвіду, висока мотивація може підвищувати швидкість та якість роботи.

6. Інші фактори. На працездатність можуть впливати також і людські фактори, які важко формалізувати. Власні переживання, життєві негаразди, хвороби, негативна ситуація у сім'ї чи суспільстві можуть значно знизити продуктивність чи взагалі тимчасово вибити людину з робочого процесу. Натомість позитивні події в інших сферах життя можуть позитивно позначатись і на роботі.

2.3 Project Management

Важливою складовою роботи над проєктом є управління проєктом (project management). PMBOK GUIDE - довідник фундаментальних практик проєктного менеджменту - подає наступне визначення цього поняття: «Управління проєктом – це застосування знань, навичок, інструментів і методів до проєктної діяльності для задоволення вимог проєкту» [34]. На великих проєктах за це відповідає окрема людина - project manager, на менших цю роль може виконувати хтось з суміжних посад. Та незалежно від назви ролі незмінними залишаються задачі й цілі, які покладаються на цю людину:

- **комунікація з командою та замовником.** Менеджер виступає проміжною ланкою між клієнтом і командою яка працює над проєктом. На нього покладаються обов'язки планування зустрічей, донесення побажань клієнта до команди, демонстрація результатів роботи команди клієнту, коригування роботи відповідно до побажань клієнта, підтримка згуртованості команди, розв'язання конфліктних ситуацій, створення звітів та іншої потрібної документації. Загально кажучи менеджер відповідає за організацію ефективної роботи, задоволення клієнта та створення комфортних умов для команди;

- **бачення повної картини проекту, розуміння цілей та передача цих знань команді**. Першочерговим етапом є правильне висвітлення наявної проблеми і постановка задачі, адже те, як буде поставлена задача, визначає те, яким способом вона буде розв'язана. При невдалій постановці цілей можна отримати продукт, який буде правильно працювати, але розв'язувати абсолютно іншу задачу [35]. Отже, важливо щоб цілі були коректно сформовані, і вся команада їх розуміла в однаковий спосіб;
- **розбиття проєкту на підзадачі.** Ключовою частиною робіт з організації проекту є структура декомпозиції робіт (work breakdown structure) - процес декомпозиції задачі на менші підзадачі, які легше планувати, виконувати та контролювати [34];
- **розподіл ресурсів**. Правильне упорядкування задач з урахуванням їх пріоритетності та наявності людських ресурсів, дозволить правильно використовувати ці ресурси. Зокрема так, щоб члени команди не залишались довгий час без роботи в очікуванні поки не з'являться (стануть доступними до виконання) задачі для них.
- **визначення термінів виконання**. Об'єктивне визначення термінів виконання, так щоб вони задовольняли не лише клієнта, а і виконавців дуже важливе, адже стиснуті терміни змушують розробників жертвувати повноцінним обговоренням та проектуванням майбутнього функціоналу, пошуком кращих варіантів рішення проблеми (замість використання вже знайомих, але можливо не оптимальних, підходів), якістю коду та тестування. Це дуже шкідливо в довгостроковій перспективі, адже з розвитком проекту зусилля розробників все більше будуть спрямовуватись не на реалізацію нових функцій, а на боротьбу з безладом, який був створений за часи ігнорування якості коду та архітектури. [36] Важливим правилом

менеджменту, що стосується цього пункту, є що «люди, що виконуватимуть роботу мають приймати участь в її плануванні»;

- **контроль над виконанням.** Задачею менеджера є моніторинг прогресу, розв'язання проблемних ситуацій, коригування процесів з урахуванням нових задач чи змін у вимогах;
- **контроль якості**. Менеджер визначає критерії та вимоги до проєкту на основі побажань клієнта та контролює наскільки поточні результати відповідають заданим критеріям.

2.4 Методологія розробки

Перед початком роботи над проєктом необхідно визначитись ще й з методологією розробки, що буде використовуватись при роботі. Різних методологій багато, і жодна з них не є універсальною - вибір має залежати безпосередньо від цілей проєкту, бюджету, термінів виконання, об'єму робіт та інших умов. У деяких випадках компанія може використовувати гібридні підходи, що поєднують декілька методологій в одному проєкті. Але в загальному випадку, всі методології поділяються на два класи - heavyweight та lightweight. Heavyweight, або як їх ще називають, традиційні методології, тяжіють до довгострокового планування та створення детальної документації, що робить цей етап проєктування дуже витратним. Типовим представником цього підходу є Waterfall, Spiral Model та Unified Process. На противагу, lightweight методології, або agile методології, оперують короткими ітеративними циклами розробки, і надають перевагу обміну знань про проєкт між членами команди, а не написанню повноцінної документації [37]. Ці методології поступово витісняють традиційні підходи та стають все більш популярніший. Прикладами такої методології можна назвати Extreme Programming, Scrum та Dynamic System Development.

Розглянемо особливості цих двох класів методологій детальніше, на прикладі їх популярних представників.

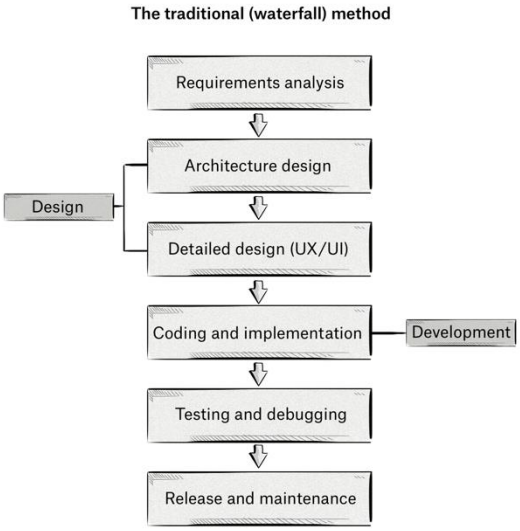

Рисунок 2 Схема етапів розробки у Waterfall методології [35]

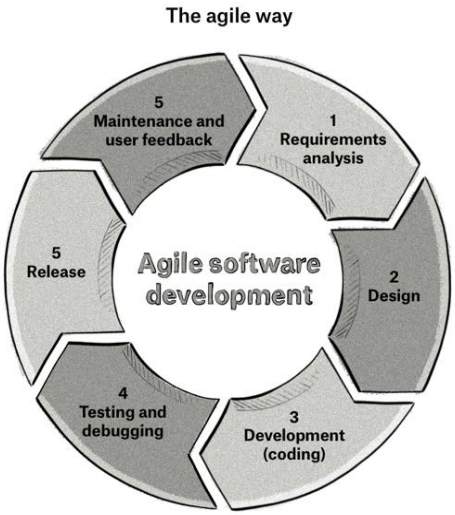

Рисунок 1 Схема етапів розробки в Agile методології [35]

Waterfall. Коротко цей підхід можна охарактеризувати як «строгий» та «лінійний». У сучасній практиці цей термін використовується як загальна назва всіх методології, у яких чітко відокремленні етапи проєкту, причому перехід до наступного етапу відбувається лише після повного закінчення попереднього і повернення на попередні кроки не передбачається (рис. 1). Кожен з цих кроків - формулювання задач проєкту, проєктування, написання коду, тестування і деплоймент - детально спланований в часі, вимоги до продукту чітко фіксуються, проєкт має детальну та вичерпну документацію. Ця методологія приваблює наявністю чіткої дисципліни в розробці і передбачуваністю результату, але відштовхує надлишковою бюрократичністю і важкістю внесення змін на фінальних етапах.

Scrum. Творці Scrum у своєму Scrum Guide характеризують його як «a lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems» [38]. У цьому підході робота організована ітераціями, які звуться спринтами (від англ. sprint).

Типово кожна така ітерація триває 2 тижні, за які команда проходить всі етапи роботи (проєктування, написання коду, тестування та деплоймент) і по завершенню представляє готову частину функціоналу (рис. 2). У цій методології робиться акцент на якісному контролі процесу роботи. Щодня учасники команди мають зустріч (у даних реаліях більшість таких зустрічей відбувається онлайн), на яких обговорюють зроблене за минулий день, плани на наступний, труднощі з якими стикнулись тощо. Крім того кожного спринта існують додаткові зустрічі. Перед початком ітерації команда разом з менеджером планують, які задачі потраплять в наступний спринт. Наприкінці проводиться зустріч, на якій всім зацікавленим демонструються доробки за цей час. Також є зустріч на якій команда має змогу оцінити невдачі та сильні сторони при роботі в останньому спринті та подумати над покращеннями. Перевага цього підходу в тому, що клієнт отримує новий функціонал поступово, малими частинами, але одразу може починати ним користуватись і отримувати зворотній зв’язок від користувачів. Якщо виявиться, що продукт не повністю задовольняє їх запити чи більше не відповідає потребам ринку, команда відносно легко може прийняти зміни вимог. Крім того, малі цикли дають змогу також і команді побачити проблеми в організації роботи і виправити їх вже в найближчих ітераціях. Загально кажучи, процеси в Scrum побудовані таким чином, щоб команда могла максимально природним чином підлаштовуватись під зміни вимог проєкту, а також постійно навчатись на своєму досвіді. [39, 40] Недоліком цієї методології є те, що майже неможливо чітко передбачити кінець проєкту, а отже спрогнозувати ресурсні та грошові витрати також не є можливим. Крім того, Scrum може здатись заскладним для людей, що звикли працювати за традиційним підходом.

Отже, heavyweight та lightweight методології абсолютно різні навіть

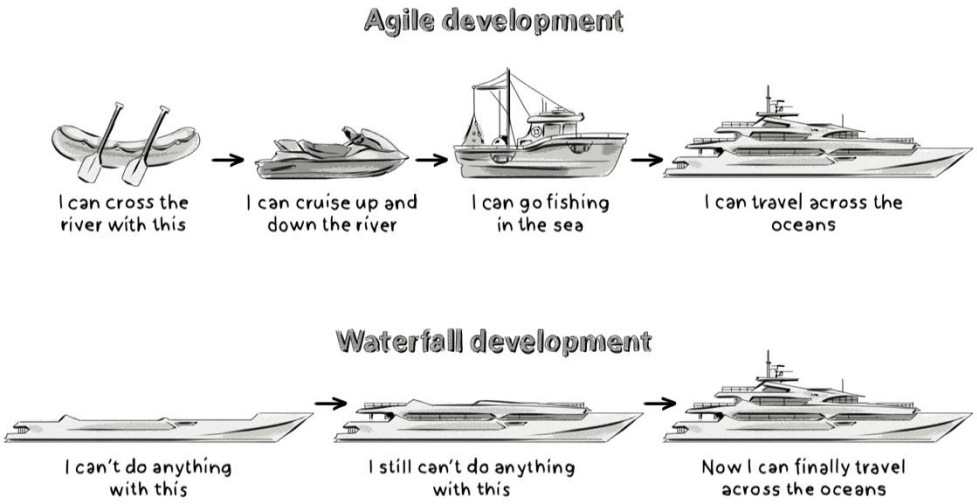

Рисунок 3 Пояснення різниці між методологіями Waterfall та Agile на прикладі будування корабля [35]

на рівні способу мислення про проєкт. Клієнти можуть обирати методологію орієнтуючись на те, що для них є важливішим - чіткість і прогнозованість чи гнучкість.

Для того щоб порівняти ці два абсолютно різні підходи, можна спробувати оцінити їх за відсотком успішно завершених проєктів. Згідно з 2013 IT Project Success Rates Survey Results при Agile підході шанси успішно закінчити проєкт на 15% вищі ніж при Waterfall (64% проти 49%). [41] Схожі результати також надає Standish Group у своєму дослідженні за 2015 рік (28% переваги) [42]. Крім того варто зауважити, що Agile підхід дозволяє зменшити ризик того, що проєкт провалиться повністю, адже вже на ранніх стадіях існує продукт, що має цінність. При Waterfall підході клієнт отримує цінність продукту лише по закінченню проєкту (рис. 3). Тобто якщо з якихось причин планування було не точним і фінансування завершилось до того як завершився проєкт, клієнт має ризик отримати напівготовий абсолютно нежиттєздатний продукт.

Висновки до розділу 2

Колективна робота групи людей це складний соціальний процес і для його якісного моделювання необхідно врахувати багато характеристик. Кожна агент - член команди має визначатись не лише вміннями й списком задач, які може виконувати, а й рисами характеру, темпераментом та багатьма іншими деталями, адже вони також мають значний вплив на роботу в команді. Окремою важливою роллю в роботі над проєктом є роль менеджера. На нього покладається багато завдань з розбиття проєкту на задачі, оцінки часу їх виконання та пріоритезації. Необхідно вирішити яким чином роль менеджера буде моделюватись у розроблюваній системі - можливо частково чи повністю ця роль покладатиметься на користувача системою. Окремо варто розглянути можливість моделювання різних методологій розробки, адже вони мають значний вплив на результат роботи команди.

РОЗДІЛ 3. ОПИС РОЗРОБЛЕНОЇ МОДЕЛІ

У рамках даної роботи була реалізована система, що моделює роботу команди розробників з використанням агентно-базованого підходу. Ціль такої системи - надати функціонал для проведення експериментів з дослідження того, як різні комбінації членів команди впливають на швидкість і якість роботи над проєктом.

Даний розділ описує технічні деталі та особливості реалізації агентів і середовища, в якому вони взаємодіють. Окремо розглядається те, які характеристики були взяті до уваги при створенні агента - учасника команди, як ці характеристики було формалізовано для використання в розрахунках, та як вони впливають на його поведінку.

3.1 Опис сутностей моделі

Для моделювання роботи команди агентів було реалізовано наступні сутності та відповідні таблиці в базі даних:

Area – позначає область/спеціалізацію (наприклад back-end, front-end, дизайн, QA тощо). Використовується як тег для позначення, якою спеціалізацією володіє агент, та до якої області відноситься завдання. Теги використовуються щоб мати можливість поєднати задачу і її потенційних виконавців.

Skill – визначає вміння та технології (наприклад SQL, Python, React, OOP). Використовується разом зі вказанням рівня (BASIC - початковий, INTERMEDIATE - середній, EXCELLENT - високий) як характеристика агента (рівень його знань) та задачі (якими знаннями і на якому рівні необхідно володіти для її розв’язання).

Person – реалізує агента – члена команди (надалі під назвою “агент” будемо мати саме екземпляр цієї сутності).

*Важливі поля***:**

- area – спеціалізація. Область значень визначений в таблиці Area. Для спрощення будемо вважати що кожен агент має лише одну спеціалізацію.
- skills_list – список вмінь. Як вже згадувалось, кожне вміння, яким володіє агент позначається як пара - вміння і рівень володіння. Наприклад Python - EXCELLENT. Виконання задачі, для якої у агента не має достатньо знань, займає більше часу ніж зазвичай, адже агенту необхідний час для навчання. При цьому такий агент може попросити допомоги у когось з потрібним досвідом, щоб прискорити роботу.
- level - загальний рівень здібностей/загальний досвід. Як описано в попередньому розділі, на практиці існує градація intern, junior, middle, senior, tech lead. Але ця градація досить умовна і в моделі рівень буде позначатись від 0.1 (початковий) до 1 (максимально високий).
- motivation – рівень мотивації, що задається на проміжку від 0.1 до 1. Високий рівень мотивації (>0.6) пришвидшує роботу над задачею, низький навпаки сповільнює.
- readiness_to_ask_for_help - готовність звертатись по допомогу - характеризує ймовірність того, що агент звернеться за допомогою до іншого агента, якщо йому не вистачає досвіду чи вмінь, щоб завершити задачу.
- willingness_to_help - готовність допомогти – вважаємо, що агент не може відмовити в допомозі іншому агенту. На проміжку від 0.1 до 1 будемо оцінювати на скільки пріоритетним для агента є допомога іншому агенту.

Task – визначає задачі, що необхідно виконати в межах проєкту. Фактично множина задач проєкту утворює собою середовище в якому працюють агенти.

Важливі поля:

- area – область задачі. Область значень визначений в таблиці Area. Задача розрахована на виконання її одним членом команди, а отже має лише одне значення area.
- importance – важливість задачі, визначена на проміжку від 0.1 до 1. Впливає на пріоритетність виконання.
- complexity – складність задачі (від 0.1 до 1). Співставляється з level агента, тобто показує, наскільки задача відповідає рівню знань члена команди. Якщо складність менша ніж рівень агента, він виконає її швидше ніж очікується. І навпаки - якщо складність вища, агенту буде необхідний додатковий час.
- type - тип задачі. Має три можливі значення: TASK - безпосередньо запит на створення нового функціоналу, BUG - запит на виправлення помилок в реалізованому раніше функціоналі, REQUEST_FOR_HELP - запит на допомогу з задачею від іншого агента (розглядається як окрема задача для полегшення пріоритезації і вибору наступного завдання до роботи).
- status – статус задачі. Може перебувати в одному з трьох станів – TO DO (очікує на виконання), IN PROGRESS (в процесі розробки), DONE (виконана).
- progress - прогрес виконання задачі - від 0 (у щойно створеної задачі) до 100% (у готової задачі). Поле потрібне для відслідковування прогресу по задачі, а також для реалізації можливості передачі задачі від одного члена команди іншому у випадку необхідності.
- story_points – кількість сторіпоінтів, у які оцінена задача. У даній роботі був використаний підхід до оцінки з використанням чисел

Фібоначчі – тобто задача може мати 1 сторіпоінт (абсолютно тривіальна, швидка у виконанні задача, наприклад виправити одруківку в тексті на сторінці сайту), 2, 3, 5 або 8 сторіпоінтів (об'ємна задача, що вимагає глибоких знань та значних зусиль). Якщо кількість сторіпоінтів не вказана при створенні задачі, їх кількість буде обчислена за формулою complexity * size, де size випадково згенероване число від 0.1 до 1. Причому, якщо складність задачі перевищує 0.3, вважаємо, що розмір задачі не може бути менший ніж

Рисунок 4 Відношення між складністю задачі і кількістю сторіпоінтів

0.4; якщо складність менша 0.3, її розмір не може бути більше 0.6. Отриманий результат масштабується на проміжок від 0 до 8 і округлюється до одного з можливих значень story points (рис. 4).

- num_of_hours – обчислюване поле - кількість годин, необхідна для розв'язання задачі умовним членом команди з достатнім рівнем знань, відповідним попереднім досвідом та нейтральною мотивацією. Обчислюється спираючись на кількість сторіпоінтів задачі. Нехай

покладемо, що приблизна відповідність сторіпоінти - години на виконання буде виглядати так:

Story points	Відповідна кількість годин
1	2
2	8
3	20
5	35
8	60

На практиці майже ніколи не траплятиметься так, що всі задачі, оцінені в однакову кількість сторіпоінтів, будуть потребувати однакову кількість годин на виконання. Тому значення num_of_hours буде обчислюватись за допомогою генерування випадкового значення в нормальному розподілі (за допомогою модуля numpy), де математичне сподівання дорівнює значенню з таблиці, а стандартне відхилення становить значення з таблиці * 0.2 (рис. 5). Результати таких розрахунків зображено на рисунка 6 і 7.

```
import numpy as np
np.random.normal(loc=expected_num_of_hours, scale=expected_num_of_hours * 0.2)
```

Рисунок 5 Код, що реалізовує підрахунок очікуваної кількості годин на виконання задачі

Це поле буде використовуватись в обчисленні кількості годин, необхідних конкретному члену команди, на виконання задачі, враховуючи його рівень знань і мотивацію.

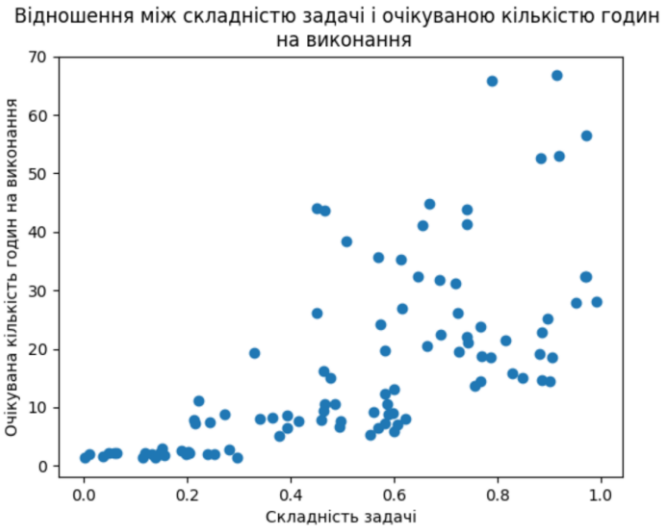

Рисунок 6 Відношення між складністю задачі і очікуваною кількістю годин на виконання

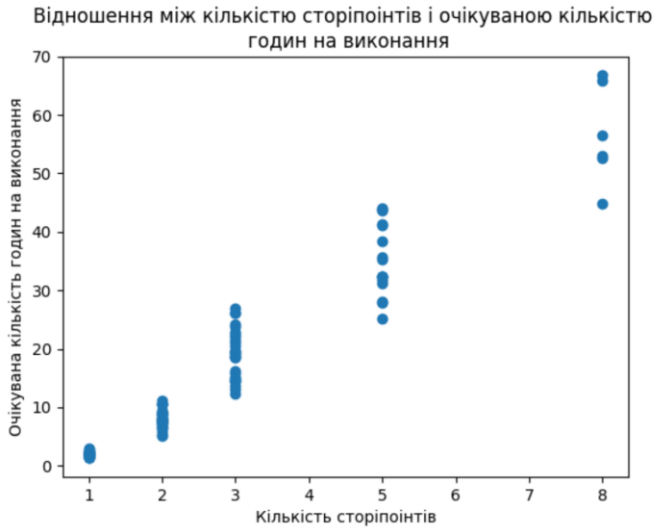

Рисунок 7 Відношення між кількістю сторіпоінтів і очікуваною кількістю годин на виконання

- skills_list - список вмінь, необхідних для розв'язання задачі. Також позначається парою - вміння і потрібний рівень володіння.
- dependency_task_list - список задач, від яких залежить поточна задача, тобто задачі, які блокують її виконання. Агент не може почати працювати над задачею, поки задачі, від яких вона залежить, не виконані.
- priority - обчислюване поле. Пріоритетність звичайної задачі дорівнює рівню її importance; баги - рівню importance функціоналу, в якому вона була виявлена; задачі з якою агента попросили допомогти, обчислюється як добуток її важливості на бажання агента допомогти іншому з задачею:

priority = task.importance * agent.willingness_to_help

На поточному етапі, для спрощення, вважаємо, що набір задач є повністю визначений на момент початку роботи і нові задачі не будуть з’являтись протягом роботи команди. Тим не менш, можуть з’являтись баги. Кожна бага підв’язана до задачі, та може бути виявлена не лише безпосередньо після закінчення задачі, а й через деякий час. Ймовірність виникнення баги вища для задач вищої складності, особливо якщо її виконував агент без відповідного рівня навичок чи з низькою мотивацією.

Наразі баги, що виникають автоматично призначаються агенту, що працював над задачею, в якій виникла бага. Очікуваний час роботи над багою генерується випадково, фактичний час - обчислюється базуючись на очікуваному часі та мотивації виконавця.

3.2 Алгоритм роботи агента

Коротко алгоритм дій агентів в середовищі можна описати так. Кожен агент працює над своєю задачею, після її закінчення, він обирає наступну задачу і працює над нею. Якщо задача заскладна для агента, або потребує забагато часу, він може попросити допомоги в іншого члена команди. Член команди, в якого просять про допомогу, допомагає, коли знаходить час для цього (задача стає пріоритетною для нього). Агент може переключатись між задачами в процесі роботи, якщо з’являється задача зі значно вищим пріоритетом. Схема описаного алгоритму зображена на рисунку 8. Розглянемо кроки алгоритму детальніше.

Вибір задачі агентом. У поточній реалізації більшість задач агент обирає для себе самостійно. Винятки - баги та прохання про допомогу. Щоб агент міг взяти задачу необхідно щоб ця задача відповідала спеціалізації агента. Також агент намагається обирати задачі, що максимально відповідають його рівню знань. Заважкі задачі займають значно більше

часу, крім того постійна робота з занадто важкими або занадто простими задачами знижує мотивацію агента. Крім того агент надає перевагу тим

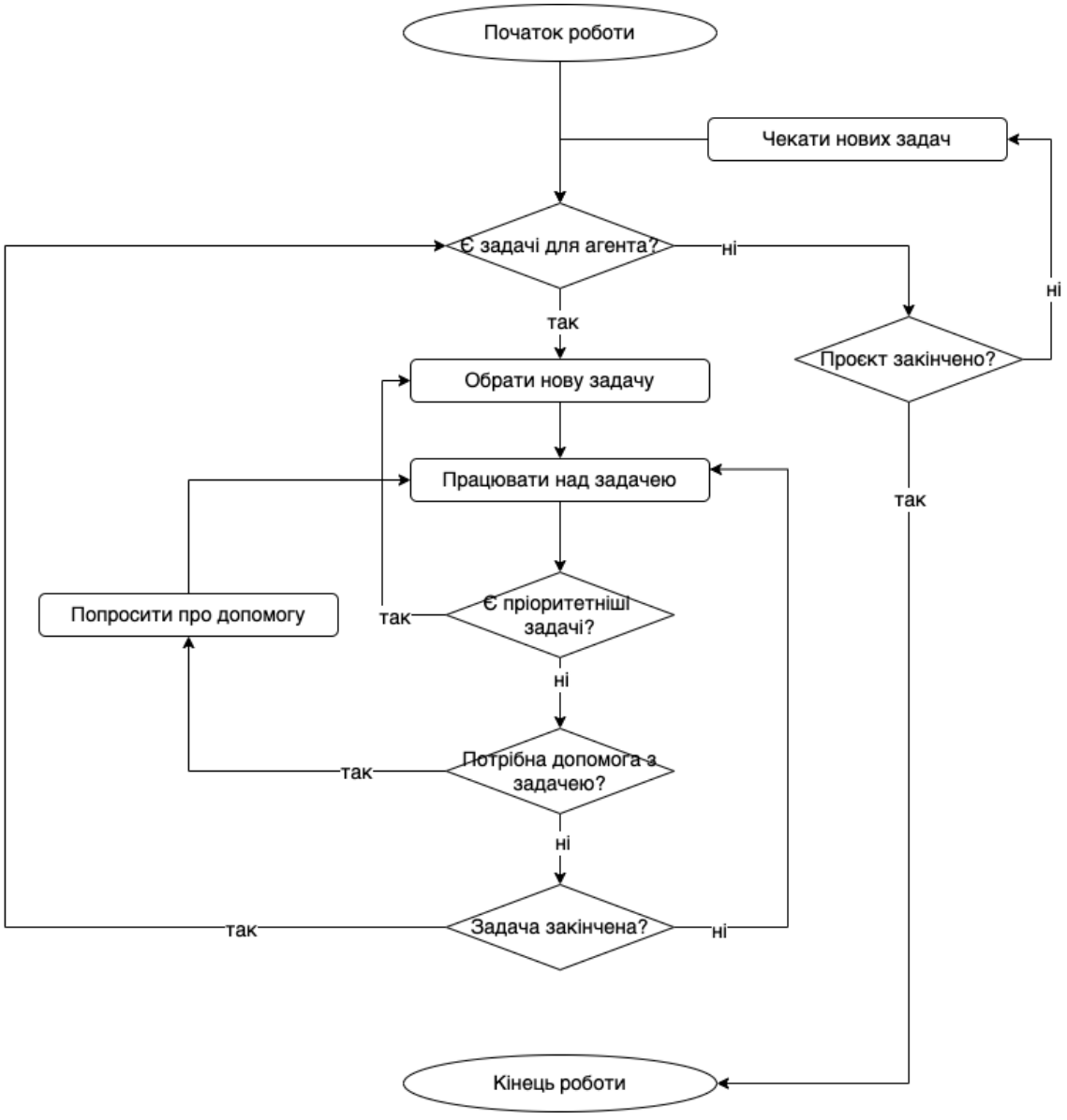

Рисунок 8 Алгоритм роботи агента

задачам, для яких у нього є релевантний досвід.

Отже, при виборі наступною задачі агент перевіряє список призначених йому задач, а також загальний список задач в статусі TO_DO та обирає найпріоритетніші з підходящих йому. Агент більш зацікавлений у виконанні задач з власного списку, тому якщо важливість найпріоритетнішої задачі з загального списку не на багато вища (не більш

ніж на 0.2) ніж важливість найпріоритетнішої задачі з власного списку, агент обере роботу над другою задачею.

Обчислення часу виконання задачі. Час виконання задачі агентом обчислюється на основі поля num_of_hours задачі, а також мотивації та рівня вмінь агента.

Перша спроба створення формули обчислення часу виконання, ґрунтованої на сторіпоінтах (рис 9), була невдала. На рисунку 10 можна

```
def task_time(self, task):
    needed_time = task.story_points * (0.5 / self.level) * (1.6 - self.motivation)
    return needed_time
```

Рисунок 9 Невдала формула обчислення часу на виконання задачі агентом

побачити розподіл часу виконання (в умовних одиниць часу) в залежності від мотивації та рівня знань. Найбільша проблема, що відразу помітна - час

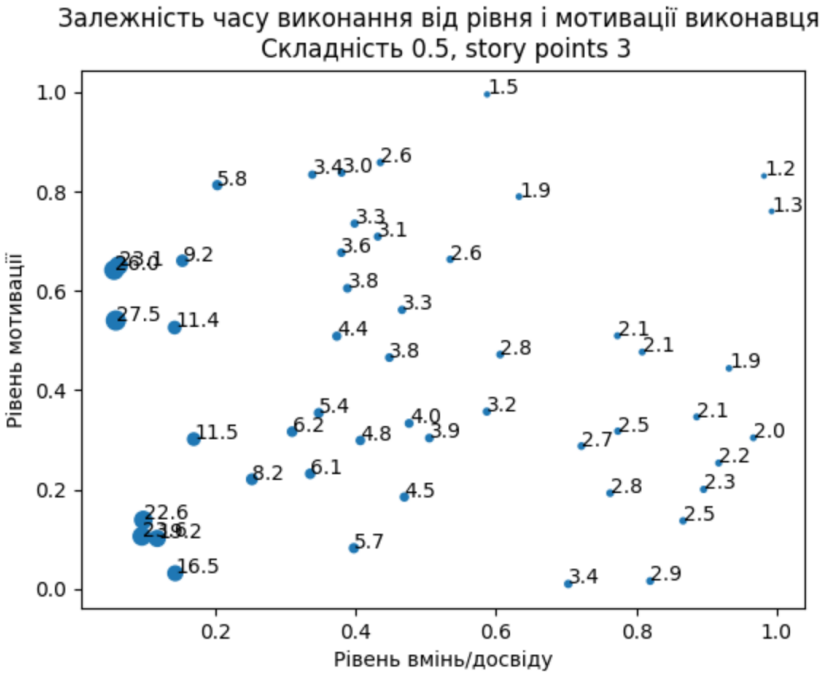

Рисунок 10 Відношення між характеристиками агента і часом виконання ним задачі, обчисленим за невдалою формулою

виконання змінюється не поступово - спостерігається величезна різниця в часі між початковими рівнями знань, і майже відсутня на вищих. Також

очевидно, що співвідношення часу між виконавцями різних рівнів

```
def task_time(self, task):
    num_of_hours = task.num_of_hours  # очікувана кількість годин на виконання
    level_diff = self.level - task.complexity  # різниця між рівнем агента і рівнем задачі
    positive_motivation = 0.6  # рівень мотивації, що вважається позитивним
    needed_time = ((1 - level_diff) * num_of_hours) * (1 - (self.motivation - positive_motivation))
    return needed_time
```

Рисунок 11 Краща формула обчислення кількості годин на виконання задачі агентом

виконання мало схожа на реальну.

Для другої формули (рис. 11) як основу для обчислень було використано не сторіпоінти, а орієнтовна кількість годин, необхідна для виконання задачі (формула розрахунку описана в розділі 3.1).

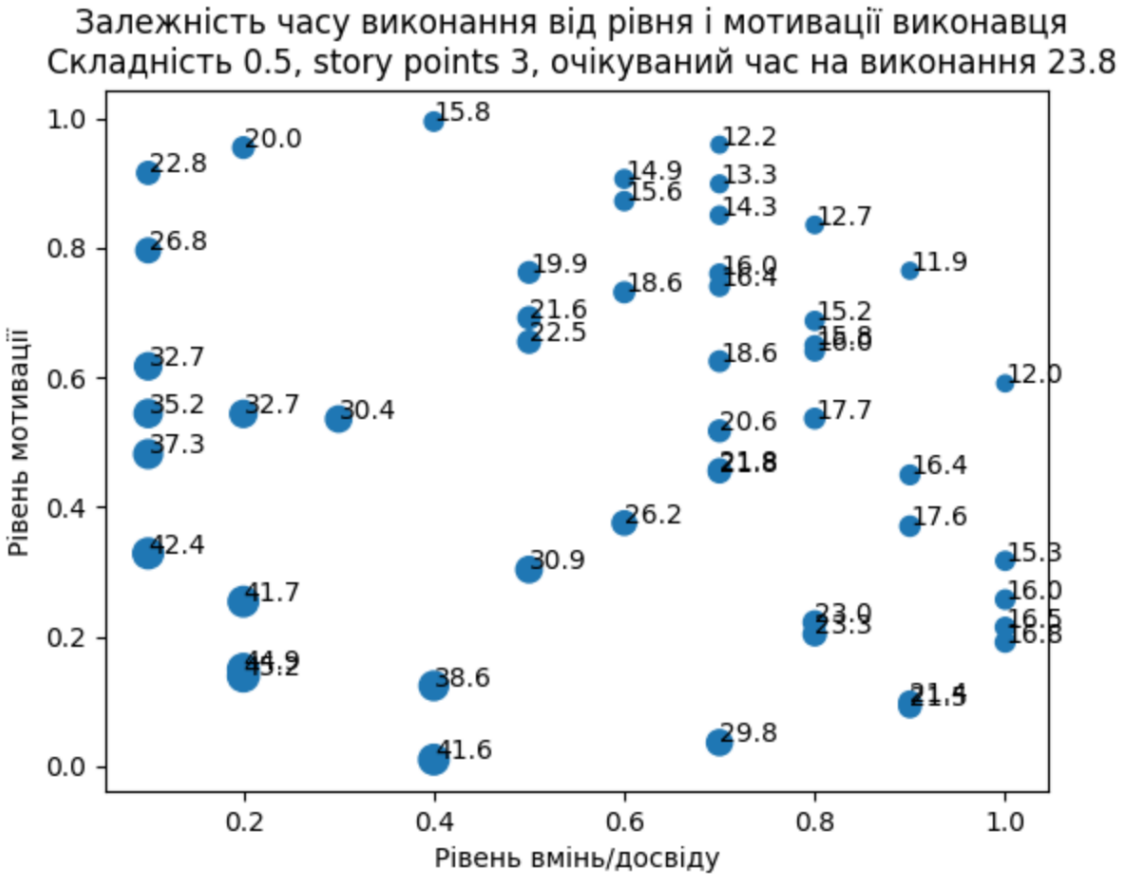

Рисунок 12 Відношення між характеристиками агента і часом виконання ним задачі, обчисленим за кращою формулою

Результат обчислень можна побачити на рисунку 12. У цьому випадку час виконання змінюється плавно при зміні рівня агента та його мотивації. Час виконання агентом з рівнем знань 0.5 і середньою мотивацією приблизно дорівнює очікуваному часу на виконання. Отже результати виглядають досить реалістично, тому ця формула буде використовуватись в моделі надалі.

Перехід між задачами. Під час роботи агент періодично (щоразу коли закінчує ще 10% роботи над задачею, якою він працює) перевіряє чи не з'явились більш пріоритетні задачі. Якщо з'явилась підходяща йому задача, важливість якої хоча б на 0.3 вища тієї, над якою він працює зараз, він зберігає прогрес поточної задачі і переходить до більш пріоритетної.

Запит про допомогу. У випадках, якщо агент працює над задачею, вищого рівня, ніж він володіє, або йому не вистачає знань якихось технологій, він може попросити про допомогу випадкового агента своєї спеціальності з рівнем знань вищим або рівним собі, щоб скоротити час на навчання і відповідно пришвидшити розробку. Щоразу після того як агент закінчує наступні 5% задачі над якої він працює, він вирішує просити про допомогу чи ні. Ймовірність того, що він її попросить визначається значенням поля readiness_to_ask_for_help. Агент, якого просять про допомогу, не обов'язково почне допомагати відразу після прохання. Поки агент очікує на іншого агента, він продовжує працювати над своєю задачею.

Допомога іншому агенту. У агента може попросити про допомогу іншого агента. Агент не можу відмовити в допомозі, але може "затягувати" з допомогою аж до моменту, коли вона вже не буде потрібна. Всі прохання про допомогу додаються до списку задач агента. Логіка обчислення пріоритетності цієї задачі для агента описано в розділі 3.1. У момент, коли агент оцінює чи є більш важливі задачі ніж та, над якою він працює зараз, він може обрати задачу, на якій очікують на його допомогу.

3.4 Використані технології та архітектура проєкту

Алгоритм, що лежить в основі моделі, було реалізовано мовою Python, версії 3.8. Дані про команду та список задач зберігаються в реляційній базі даних Postrgres. Для полегшення маніпуляції з даними було використано інструмент SQLAlchemy, що надає зручну ORM, яка дозволяє зіставити таблиці в базі даних та реалізовані в коді класи. Створення та керування міграціями забезпечується інструментом Alembic, розробленим спеціально для SQLAlchemy. Також для деяких обчислень, як от генерації нормального розподілу, використовується бібліотека NumPy.

Структура проєкту зображена на рисунку 13. Основний алгоритм, описаний в попередньому розділі, зберігається в папці simulation. У папці migration зберігаються налаштування та шаблони, що визначають логіку створення міграції, та самі міграції. Тести, що були необхідні для перевірки релевантності формул та код для побудови графіків розподілів, наведених вище, знаходяться в папці tests.

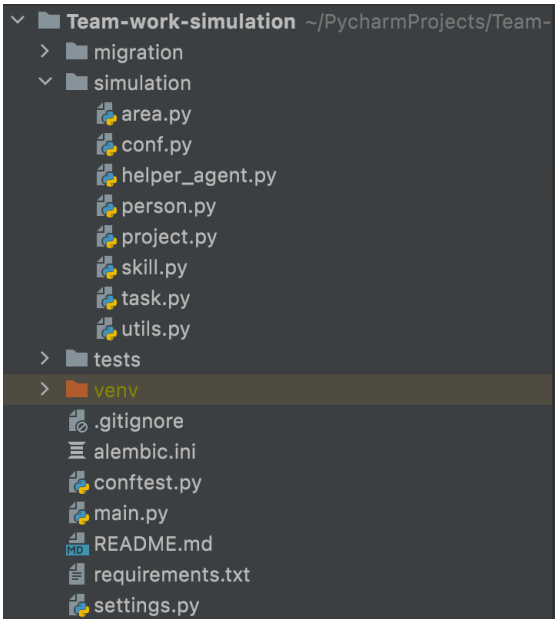

Рисунок 13 Структура проєкту

Агенти мають працювати над задачами паралельно. Оскільки через особливості керування пам’яттю Python однопоточний, для реалізації паралельної роботи було використано модуль concurrent.futures, що надає високорівневий інтерфейс для асинхронного запуску програм. Цей модуль пропонує до використання два класи: ThreadPoolExecutor, що дозволяє запускати програму в декількох потоках одного процесу, та ProcessPoolExecutor, що дозволяє запускати програму в декількох процесах на різних ядрах. Обидва класи реалізовують однаковий інтерфейс, визначений в абстрактному класі Executor. Під час розробки було протестовано обидва ці класи і обрано ProcessPoolExecutor.

Тести моделі проводились на MacBook Pro (16-inch, 2019) з процесором 2,6 GHz 6-Core Intel Core i7 та 16 GB оперативної пам’яті.

3.5 Напрямки для вдосконалення

Розроблена модель надає можливість імітувати роботу команди розробників над задачею. Агенти здатні обирати задачі, працювати над ними, просити допомоги та допомагати іншим. Тим не менш, це лише перша ітерація розробки, що має обмеження, а отже напрямки для покращення:

- додати більше властивостей агента, що б характеризували його з точки зору особистості, а не вмінь (наприклад рівень відповідальності, когнітивний стиль, темперамент, швидкість навчання тощо); зробити щоб рішення агентів про те з ким і як взаємодіяти більше залежало від їх характеристик.
- зробити можливість додавати агенту декілька спеціалізацій, щоб модель більш відповідала реальному життю;

- зробити щоб у процесі виконання задач агент навчався, тобто підвищував значення рівня компетенції, а також додавав нові вміння до списку вмінь;
- провести експерименти, щоб перевірити наведені в роботі формули на релевантність;

Інший напрямок для продовження роботи над темою – проведення досліджень з використанням створеної моделі та перевірка отриманих теорій на практиці з використанням соціологічних інструментів.

Висновки до розділу 3

У розділі описано алгоритм, закладений в основу розробленої моделі, розглянуто архітектуру проєкту, використані технології та технічні характеристики комп’ютера, на якому проводились експерименти. Також було розглянуто напрямки покращення моделі, майбутні доробки й можливості для досліджень.

Висновки

Дана робота була присвячена дослідженню соціального явища колективної діяльності та можливості його моделювання з допомогою агентно-базованого підходу.

У результаті було розроблено модель, яка імітує роботу команди розробників над проєктом. Така модель може бути використана для проведення соціологічних досліджень з метою кращого розуміння процесів, що відбуваються в середині команди.

Розроблена модель має значну кількість можливостей для покращення, та є лише першою ітерацією в дослідженні. Наступним важливим кроком є співпраця з соціологами та проведення експериментів для підтвердження релевантності результатів моделювання.

Список використаної літератури

1. Wagner, Caroline S. and Loet Leydesdorff. (2005) Globalisation in the network of science in 2005: The diffusion of international collaboration and the formation of a core group.
2. M. M. Perišić, T. Martinec, M. Štorga and T. Kanduč. (2016) Agent-based simulation framework to support management of team performing development activities.
3. Bonabeau, E. (2001) Agent-based modeling: methods and techniques for simulating human systems. In Proc. National Academy of Sciences 99(3): 7280-7287.
4. Wooldridge, M. and Jennings, N.R. (1995) Intelligent Agents Theory and Practice. The Knowledge Engineering Review, 10, 115-152.
5. Wooldridge, M. (2001). An introduction to multi-agent systems. Chichester, UK: John Wiley & Sons.
6. Yoav Shoham. (1993) Agent-oriented programming https://doi.org/10.1016/0004-3702(93)90034-9 (18)
7. The Alan Turing Institute. Електронний ресурс. Доступ 25.06.2022 https://www.turing.ac.uk/research/interest-groups/multi-agent-systems
8. Stephen D. J. McArthur, Euan M. Davidson, Victoria M. Catterson, Aris L. Dimeas, Nikos D. Hatziargyriou, Ferdinanda Ponci, and Toshihisa Funabashi. (2007) Multi-Agent Systems for Power Engineering Applications—Part I: Concepts, Approaches, and Technical Challenges.
9. Arun Sukumaran Nair, Tareq Hossen, Mitch Campion, Daisy Flora Selvaraj, Neena Goveas, Naima Kaabouch and Prakash Ranganathan. (2018) Multi-Agent Systems for Resource Allocation and Scheduling in a Smart Grid.
10. D. Koesrindartoto, S. Junjie, and L. Tesfatsion. (2005) An agent-based computational laboratory for testing the economic reliability of wholesale

power market designs in Proc. IEEE Power Eng. Soc. General Meeting, 2005, Jun. 2005, pp. 931–936.

11. Wulfrano Arturo, Luna-Ramirez, and Maria Fasli. (2018) Bridging the Gap between ABM and MAS: A Disaster-Rescue Simulation Using Jason and NetLogo.
12. Jose Simmonds; Juan A. Gómez; Agapito Ledezma. (2020) The role of agent-based modeling and multi-agent systems in flood-based hydrological problems: a brief review.
13. C. C. Liu, J. Jung, G. T. Heydt, V. Vittal, and A. G. Phadke. (2000) The strategic power infrastructure defense (SPID) system. A conceptual design," IEEE Control Syst. Mag., vol. 20, no. 4, pp. 40–52, Aug. 2000.
14. A. Halinka; P. Rzepka; M. Szablicki. (2015) Agent model of multi-agent system for area power system protection. 10.1109/MEPS.2015.7477185
15. Zubair A.Baig. (2012) Multi-agent systems for protecting critical infrastructures: A survey.
16. Roland Juchmes, Pierre Leclercq and Sleiman Azar. (2004) A Multi-Agent System for the Interpretation of Architectural Sketches.
17. Mariacristina Roscia; Michela Longo; George Cristian Lazaroiu (2013) Smart City by multi-agent systems.
18. Denise Pumain (2011) Multi-agent System Modelling for Urban Systems: The Series of SIMPOP Models.
19. J. Kruse, A. M. Connor, S. Marks (2021) An Interactive Multi-Agent System for Game Design.
20. B. Burmeister; A. Haddadi; G. Matylis. (1997) Application of multi-agent systems in traffic and transportation.
21. The Charles E. Via Jr Department of Civil and Environmental Engineering, Virginia Polytechnic Institute and State University, 7054 Haycock Road, Falls Church, VA 22043, USA. (2010) Transport modeling by multi-agent systems: a swarm intelligence approach.

22. António S. Gonçalves, Armanda Rodrigues, Luís Correia. (2004) MULTI-AGENT SIMULATION WITHIN GEOGRAPHIC INFORMATION SYSTEMS.
23. Christof Baeijs, Yves Demazeau, Luis Alvares. (2005) SIGMA: Application of Multi-Agent Systems to Cartographic Generalization.
24. L. Cingiser DiPippo; V. Fay-Wolfe; L. Nair; E. Hodys; O. Uvarov. (2002) A real-time multi-agent system architecture for e-commerce applications.
25. Niazi, Muaz; Hussain, Amir (2011). Agent-based Computing from Multi-agent Systems to Agent-Based Models: A Visual Survey (PDF). Scientometrics (Springer) 89 (2): 479–499. doi:10.1007/s11192-011-0468-9. Архів оригіналу за 12 жовтня 2013.
26. Federico Bianchi and Flaminio Squazzoni. (2015) Agent-based models in sociology. WIREs Comput Stat 2015, 7:284–306. doi: 10.1002/wics.1356.
27. Yoav Bergner Jessica J. Andrews Mengxiao Zhu Joseph E. Gonzales. (2016) Agent-Based Modeling of Collaborative Problem Solving.
28. Samuel Lapp, Kathryn Jablokow and Christopher McComb. (2019) KABOOM. an agent-based model for simulating cognitive style in team problem solving.
29. Witkin, Moore, Goodenough, and Cox (1977) Field-Dependent and Field-Independent Cognitive Styles and Their Educational Implications.
30. Zoltán Dörnyei, Ema Ushioda. (2011) Teaching and Researching: Motivation.
31. Project Managent Institute. What is Project Management? Електронний ресурс. Доступ 26.06.2022 https://www.pmi.org/about/learn-about-pmi/what-is-project-management
32. Dan Radigan. Story points and estimation. Електронний ресурс. Доступ 26.06.2022 https://www.atlassian.com/agile/project-management/estimation

33. University of Waterloo. Task Estimating. Електронний ресурс. Доступ 26.06.2022 https://uwaterloo.ca/ist-project-management-office/methodology/project-management/planning/project-schedule/task-estimating
34. A Guide to the project Management Body of Knowledge (PMBOK® GUIDE) Sixth Edition.
35. Fundamentals of Project Management. Електронний ресурс. Доступ 26.06.2022 https://books.google.com.ua/books?hl=uk&lr=&id=Vy58DAAAQBAJ&oi=fnd&pg=PR2&dq=project+management&ots=7phbNZZGY7&sig=1JqEGNeV5xcm85flRQazqhn-B0I&redir_esc=y#v=onepage&q=project%20management&f=false
36. Robert Martin. (2019) Clean Architecture: A Craftsman's Guide to Software Structure and Design.
37. Abhinav Chandrababu. (2005) A Comparison between Agile and Traditional Software Development Methodologies.
38. The 2020 Scrum Guide. Електронний ресурс. Доступ 26.06.2022. https://scrumguides.org/scrum-guide.html
39. Claire Drumond. What is Scrum? Електронний ресурс. Доступ 26.06.2022. https://www.atlassian.com/agile/scrum
40. Michael Sweeney. (2022) Agile vs Waterfall: Which Method is More Successful? Електронний ресурс. Доступ 26.06.2022. https://clearcode.cc/blog/agile-vs-waterfall-method/
41. 2013 IT Project Success Rates Survey Results. Електронний ресурс. Доступ 26.06.2022. https://www.ambysoft.com/surveys/success2013.html
42. 2015 CHAOS report from the Standish Group. Електронний ресурс. Доступ 26.06.2022. https://www.infoq.com/articles/standish-chaos-2015/