Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Метод управління вимогами в Agile-проєктах з розробки програмного забезпечення

Виконав(ла): студент(ка) 6 курсу, групи СНд-2
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

(підпис) Тарасовський Ю.В. (прізвище та ініціали)

Керівник (підпис) Савків В.Б. (прізвище та ініціали)

Нормоконтроль (підпис) Мацюк О.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Коноваленко І.В. (прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«21» вересня 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

студенту Тарасовський Юрій Вікторович
(прізвище, ім'я, по батькові)

1. Тема роботи **Метод управління вимогами в Agile-проєктах з розробки програмного забезпечення**

Керівник роботи **к.т.н., доц. Савків В.Б.**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*28*» *жовтня* 2021 року № *4/7-912*

2. Термін подання студентом завершеної роботи 20 грудня 2021 р.

3. Вихідні дані до роботи Літературні джерела з тематики роботи

4. Зміст роботи (перелік питань, які потрібно розробити)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Приймак М.В., д.т.н., проф.		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викл.		

7. Дата видачі завдання 21 вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	21.09.21-27.09.21	*Виконано*
2.	Підбір наукових джерел по темі роботи	28.09.21-04.10.21	*Виконано*
3.	Переклад та опрацювання наукових джерел	05.10.21-11.10.21	*Виконано*
	по темі кваліфікаційної роботи		
4.	Виконання дослідження щодо огляду атак на	12.10.21-18.10.21	*Виконано*
	комп'ютерні системи		
5.	Оформлення першого розділу	19.10.21-25.10.21	*Виконано*
6.	Оформлення другого розділу	26.10.21-01.11.21	*Виконано*
7.	Оформлення третого розділу	02.11.21-08.11.21	*Виконано*
8.	Виконання завдання до підрозділу «Охорона праці»	09.11.21-15.11.21	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	16.11.21-22.11.21	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	23.11.21-29.11.21	*Виконано*
11.	Нормоконтроль	30.11.21-05.12.21	*Виконано*
12.	Перевірка на плагіат	05.12.21	*Виконано*
13.	Попередній захист кваліфікаційної роботи	14.12.21	*Виконано*
14.	Захист кваліфікаційної роботи	20.12.2121	

Студент _____ Тарасовський Ю.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Савків В.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

"Метод управління вимогами в Agile-проєктах з розробки програмного забезпечення" // Тарасовський Юрій Вікторович // Тернопільський національний технічний університет ім. Івана Пулюя, центр перепідготовки і післядипломної освіти, кафедра комп'ютерних наук, група СНд-2 // Тернопіль, 2021 // с. – 60, рис. – 14, табл. – 7, джерел – 61.

Ключові слова: РОЗРОБКА ПЗ, AGILE, УПРАВЛІННЯ ПРОЄКТОМ, ГНУЧКА РОЗРОБКА, ПОРІВНЯННЯ.

Agile методи розробки програмного забезпечення характеризуються як гнучкі та легко адаптовані. Потреба не відставати від кількох високопріоритетних проектів і скорочення термінів виходу на ринок готових програмних продуктів може пояснити їх зростаючу популярність. Оскільки методи Agile дозволяють вносити зміни протягом усього процесу розробки, вони також створюють ймовірність впливу на якість програмного забезпечення в будь-який момент. У цій роботі розглядається процес розробки вимог, що виконується за допомогою методу Agile, з точки зору його подібності та відмінності від інженерії вимог, що виконується за допомогою більш традиційного водоспадного методу.

ANNOTATION

"The method for requirements management in software Agile projects " // Yurii Tarasovskyi // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Computer Science Department, group СНд-2 // Ternopil, 2021 // p. – 60, fig. – 14, tables. – 7, ref. – 61.

Keywords: SOFTWARE DEVELOPMENT, AGILE, PROJECT MANAGEMENT, FLEXIBLE DEVELOPMENT, COMPARISON

Agile software development methods are characterized as flexible and easily adapted. The need to keep up with several high-priority projects and shortening the time to market of finished software products may explain their growing popularity. Because Agile methods allow to make changes throughout the development process, they are also likely to affect the quality of the software at any time. This paper considers the process of requirements development performed using the Agile method, in terms of its similarities and differences from the requirements engineering performed using the more traditional waterfall method.

ЗМІСТ

ВСТУП

Актуальність теми.

У сучасному світі швидкість розробки (створення) програмного забезпечення є одним із найважливіших факторів, щоб позначити його як успішне чи провальне. Цей фактор обумовлений постійною потребою клієнта в змінах, яка була втілена у відносно новій методології розробки програмного забезпечення під назвою «Agile». Agile розробка програмного забезпечення стала одним з найпопулярніших підходів. Цей метод має багато практичних реалізацій, усі вони зосереджені навколо задоволення клієнтів, постійної зміни вимог, повторюваних циклів або випусків програмного забезпечення та глибокої участі та внеску бізнес-користувачів. Навпаки, розробка вимог або аналіз вимог в основному займає багато часу. Традиційний процес збору, аналізу, перевірки та документування вимог є важливим для будь-якої розробки програмного забезпечення.

На даний момент ми можемо зрозуміти, що Agile має за пріоритет швидку доставку продукту замовнику, а типові для гнучких методів розробки зустрічі з веденням мінімальної кількості документації не дають можливість сказати, що розробка вимог буде виконана належним чином з врахування можливості передати ці зібрані вимоги та зібраний досвід іншим працівникам компанії пізніше. Несумісність очевидна, і мета цієї роботи – зменшити цей розрив між Agile та інженерією вимог. Це буде досягнуто шляхом виявлення критичних проблем, які впливають на процес розробки вимог у Agile, та надання деяких найсучасніших практик, які можуть призвести до покращення аналізу вимог у Agile.

Проблемами збору і інженерії вимог до програмних продуктів займались також в ТНТУ, наприклад роботи [56], [57].

Мета роботи. Основна мета цієї роботи полягає в тому, щоб надати емпіричні дані з реальних випадків, які ілюструють, що Agile-методи суттєво впливають на якість програмного забезпечення, включаючи можливість більшої кількості дефектів через погане виявлення нефункціональних вимог.

Тому були поставлені такі задачі:

1. Чи є суттєва різниця у кількості дефектів, що виникають у методах Agile та Waterfall?

2. Чи існує суттєва різниця у кількості пов'язаних із нефункціональними (Nonfunctional Requirements – NFR) чи функціональними (Functional Requirements – FR) вимогами дефектів, що виникають у методах Agile та Waterfall?.

Об'єкт дослідження: процеси збору вимог до програмного продукту

Предмет дослідження: ітераційні (гнучкі) моделі життєвого циклу програмних продуктів.

Практична цінність роботи.

Незважаючи на популярність методу Agile в реальному світі, його переваги та ризики не були емпірично перевірені за допомогою реальних випадків, які виникли в реальних життєвих ситуаціях. До цих пір більшість робіт, присвячених методам Agile, покладалися на оцінки, зроблені розробниками, які використовують цей метод, і лише їхній досвід. Дослідження пропонує емпіричні докази впливу методів Agile на якість програмного забезпечення.

Оскільки більшість організацій прагнуть запровадити ефективні системи/процедури щодо інженерії вимог (Requirements Engineering – RE) та програмного забезпечення, ці висновки можуть бути корисними для компаній, які намагаються краще зрозуміти складні порівняння між методами Agile та Waterfall, вирішуючи, чи використовувати методи Agile.

Апробація результатів та особистий внесок здобувача. Основні положення роботи доповідались, розглядались та обговорювались на науковій конференції Тернопільського національного технічного університету. Результати кваліфікаційної роботи опубліковані у тезах студентської наукової конференції, яка проводились у ТНТУ.

1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ ПО ТЕМАТИЦІ РОБОТИ

У цьому розділі наведено довідкову інформацію про два методи розробки програмного забезпечення (Agile і Waterfall), які досліджуються в цьому дослідженні. Спочатку визначається метод розробки програмного забезпечення, а потім описуються характеристики та переваги кожного з них. Далі вводяться визначення інженерії вимог та забезпечення якості програмного забезпечення та обговорюються їх застосування в обох методах. Протягом усієї глави порівняння методів Agile і Waterfall виявляє їх схожість і відмінності в кожній області (наприклад, переваги, розробка вимог і процедури забезпечення якості).

1.1 Методи розробки програмного забезпечення

Протягом багатьох років було впроваджено багато систематичних, регламентованих та кількісних методів розробки програмного забезпечення. Кожен з цих методів включає в себе кілька етапів і різноманітні види діяльності. Наприклад, проектування, рефакторинг, повторне використання, реінжиніринг та обслуговування є одними із звичайних видів діяльності, які використовуються для створення програмних рішень [31]. Метою цієї діяльності є розробка програмного забезпечення, яке задовольняє потреби клієнтів.

За останні 50 років було впроваджено кілька методів для підвищення успіху в розробці програмного забезпечення. Деякі з поширених методів включають водоспад, прототипування, ітерацію та інкрементну розробку, спіраль, швидку розробка додатків, екстремальне програмування та Agile. Вибір методу розробки програмного забезпечення є важливим у будь-якому програмному проекті, оскільки сильні та слабкі сторони методу впливають на задоволення замовника [20].

Деякі дослідники припускають, що вибір конкретного методу пов’язаний з низкою ситуаційних факторів, включаючи організаційні, проектні та командні фактори [36], а отже, один метод не підходить для всіх проектів розробки програмного забезпечення. Рішення про використання конкретного методу може

бути засноване на літературі та стандартах узгодженості в галузі [51]. У наступному розділі описані характеристики та переваги методів розробки Agile та Waterfall.

1.1.1 Метод Agile Development

Метод Agile Development з'явився в результаті збору сімнадцяти представників індустрії розробки програмного забезпечення в Сноуберді, штат Юта, у 2001 році [2]. Їх метою було просування інноваційних підходів до розробки програмного забезпечення, які дозволять організаціям швидко реагувати та адаптуватися до мінливих вимог і технологій. У своєму маніфесті Agile [2] вони визначили такі чотири пріоритети:

a) особи та взаємодія над процесами та інструментами,

b) робоче програмне забезпечення над вичерпною документацією,

c) співпраця з клієнтом над узгодженням контрактів і

d) відповідь на змінювати, а не слідувати плану.

Маніфест заснований на дванадцяти принципах, спрямованих на досягнення задоволеності клієнтів шляхом частої доставки працюючого програмного забезпечення та перебування в команді, що самоорганізується. Існує багато типів методів Agile, таких як екстремальне програмування, Scrum, розробка на основі функцій, метод розробки динамічної системи, адаптивна розробка програмного забезпечення, а також Lean розробка програмного забезпечення [4] [9] [50] [35]. Спільним для всіх методів є поділ потреб клієнтів на кілька циклів випуску, які представлені меншими частинами відповідно до їхньої бізнес-цінності [52] [20]. Ці методи включають деякі з найбільш впізнаваних факторів якості, такі як: економічність, ефективність, розширюваність, ремонтопридатність, портативність, можливість повторного використання та надійність [28].

Загалом, Agile заохочує команди розробників програмного забезпечення залишатися максимально гнучкими, працюючи над досягненням спільної мети – досягнення успішного результату.

Сучасне динамічне бізнес-середовище все частіше обирає методи Agile над іншими методами через його переваги [9] [20] [25] [31] [36] [45] [51]. Завдяки

здатності обробляти нестабільні вимоги протягом життєвого циклу розробки програмного забезпечення та можливості виконувати роботу в більш короткі терміни, гнучкі методи отримали визнання [32]. Основними перевагами методів Agile є: 1) зниження ризику та 2) збільшення доходу.

У методах Agile ризик збою зменшується завдяки ітераційному підходу з частим безперервним плануванням і зворотним зв'язком. Хоча команда повинна залишатися зосередженою на забезпеченні узгодженої частини функцій продукту під час кожної ітерації, є можливості постійно вдосконалювати й змінювати пріоритети загального відставання продукту. Нові або змінені елементи відставання можна запланувати на наступну ітерацію і надати можливість внести зміни протягом кількох тижнів. Незалежно від розміру проекту, відповідь клієнта швидка та часта. Хоча, за словами Standish Group, метод Agile вважається одним із факторів успіху малих проектів, оскільки він втілює філософію малих проектів [46], більші проекти також можна розбити на менші проекти та виконуватися в ітераційному процесі. .

Оскільки ризик зменшується, дохід також збільшується за рахунок чіткого розуміння пріоритетів. Він підтримує зниження накладних витрат на створення, повторне виконання та підтримку вимог [25] [31] і допомагає отримати більший дохід. Активна участь користувача або представників бізнесу в процесі розробки програмного забезпечення створює позитивні робочі відносини та більш точне розуміння цілей проекту. Наприклад, опитування Liu et al. [27] 377 учасників із китайських компаній-розробників програмного забезпечення та дослідників показало, що трьома причинами серйозних збоїв було нечітке розуміння систем, нерозуміння постійних змін і недостатня взаємодія з клієнтами. Окрім частішої доставки порцій продуктів, метод Agile підтримував робочі функції як найважливіший показник прогресу протягом життєвого циклу проекту, щоб принести більше доходу за коротші терміни [7]. Наприклад, зміна пріоритетів, додавання та модифікація існуючих вимог є поширеними в Agile, оскільки розглядається як внесок у мінімізацію витрат [7] [25]. Agile забезпечує ефективність процесу, щоб максимально використовувати ресурси та максимізувати прибуток.

1.1.2 Водоспадний метод розробки

Протягом багатьох років багато методів розробки програмного забезпечення були вдосконалені та іменовані як «традиційні» методи. Одним із найстаріших традиційних методів є водоспадний, який вперше був описаний у [40] і досі широко практикується як у малих, так і у великих проектах [3]. Метод розробки Waterfall складається з послідовних етапів, включаючи аналіз вимог, розробку програмного забезпечення, впровадження, тестування, інтеграцію, розгортання та обслуговування. Як правило, у цій моделі діяльність розглядається як більш стабільна низхідна, і більше уваги приділяється плануванню, розкладам часу, цільовим датам, бюджетам і одночасному впровадженню всієї системи. Ця модель визначає пріоритет, щоб команда розуміла всі вимоги проекту до початку, оскільки плани не переоцінюються під час процесу розробки.

Історично склалося так, що під водоспадом було реалізовано багато успішних проектів. Він широко використовувався як у малих, так і у великих проектах [3], і було повідомлено про багато переваг.

Деякі з ключових переваг Waterfall: підвищення ефективності за допомогою стабільного визначення проекту та легшого керування завдяки жорсткості кожної фази. Водоспадний метод відомий своїми більш конкретними та повними вимогами, і ці характеристики роблять цей підхід більш стабільним. Як згадувалося раніше, перший із п’яти етапів цього методу – це аналіз вимог. У цьому методі вважається, що витрачання більше часу на початку циклу може призвести до більшого успіху на пізніх етапах. Крім того, керувати методом водоспаду легше завдяки лінійному підходу.

1.1.3 Порівняння методів Agile та Waterfall

Методи Agile і Waterfall відрізняються за типами діяльності та за тим, коли вони використовуються в процесі розробки. За даними [32], підхід Agile повертає традиційний програмний процес «набік». У той час як традиційні методи розробки, керовані планом, такі як Waterfall, використовують окремі та лінійні етапи, такі як

аналіз вимог, проектування системи, впровадження, тестування та обслуговування, методи Agile включають процеси, що перекриваються [3] [32].

На рисунку 1.1 [32] показано, що у Waterfall здійснюються такі види діяльності, як «Аналіз та визначення вимог», «Проектування системи та програмного забезпечення», «Впровадження та тестування модулів», «Інтеграція та тестування системи» та «Експлуатація та технічне обслуговування». один за одним у послідовному порядку. Однак у методах Agile після «створення історій користувачів» і «планування випуску» такі дії, як «планування ітерації», «створення модульного тесту», «розробка коду», «безперервна інтеграція», «приймальні випробування» проводяться одночасно.

1.2 Інженерія вимог (RE)

Інженерія вимог — це наукомістка діяльність, яка включає виявлення, аналіз, документування, перевірку та керування вимогами до програмного забезпечення [4] [18] [25] [31]. За словами автора в [49], інженерія програамних вимог (Requirements Engineering – RE) забезпечує відповідний механізм, щоб зрозуміти, чого хоче клієнт [18], і допомагає проаналізувати його потреби, щоб можна було визначити функціональність програмного забезпечення та обговорити прийняті рішення. Згідно з Міжнародною інженерною радою вимог, RE є формалізованим підходом, який виконується шляхом документування та систематичного управління потребами зацікавлених сторін [46].

1.2.1 Типи RE

Вимоги можна розділити на дві категорії: нефункціональні вимоги (NFR) і функціональні вимоги (FR). NFR також відомі як вимоги до якості та відносяться до критеріїв, які можна використовувати для кваліфікації роботи системи та визначення того, як програмне забезпечення має вести себе в будь-який момент часу. Деякі з NFR

включають вимоги щодо надійності, безпеки, точності, безпеки, продуктивності, зовнішнього вигляду та відчуття, а також організаційні, культурні та політичні вимоги [13].

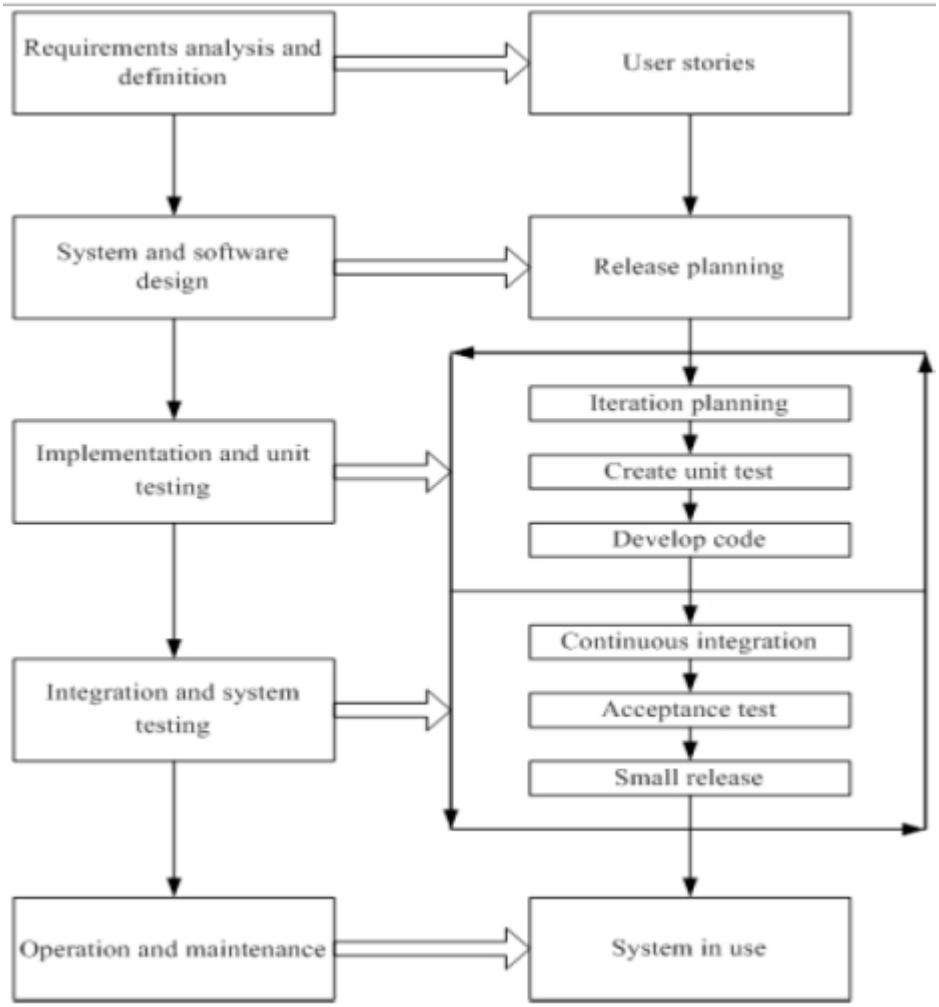

Рисунок 1.1 – Порівняння життєвих циклів Agile і Waterfall

З іншого боку, FR відноситься до певної поведінки, яка пов’язана з функцією програмного забезпечення, щоб визначити, які функції та функції воно повинно забезпечувати. Наприклад, FR може включати обчислення, маніпулювання даними та будь-які конкретні бізнес-правила, що стосуються процесів. У наступному розділі коротко обговорюється, як RE виконується в методах Agile і Waterfall.

1.2.2 RE в Agile

У Agile RE досягається завдяки безперервній співпраці, а збір вимог, розробка та тестування можуть відбуватися одночасно. Зараз багато організацій, що займаються розробкою програмного забезпечення, мають справу з вимогами, які мають тенденцію швидко розвиватися і застарівати ще до завершення проекту [5]. Отже, потреба в безперервному РЕ вважається необхідною для успішної доставки в цих випадках. Автори у [18] вказали, що останні дослідження показали, що інженери з вимог віддають перевагу коротким вимогам. У польовому дослідженні компанії з 26 членами, які практикують Agile, команди поступово вловлювали вимоги та спілкувалися усно, а не використовували вичерпний документ для узагальнення знань про вимоги [18]. У Agile бізнес-вимоги висвітлюються та документуються у вигляді історій користувачів, які зображують з точки зору користувача [46].

Ці історії користувачів використовуються як основна одиниця роботи і продовжують зростати протягом життєвого циклу проекту. Автори [25] провели якісне дослідження RE-практик у 16 організаціях, які практикують Agile, і повідомили, що його ключові практики включали: спілкування віч-на-віч, ітераційну інженерію вимог, екстремальне визначення пріоритетів, постійне планування, розробку на основі тестування, огляди. , і тести. Хоча Agile-розробники наполягають на безпосередньому залученні користувачів і доставці «версій продуктів» у коротких кроках, потреба у вимогах є дуже важливою [35]. Перевірка цих вимог також продовжується під час RE в Agile.

1.2.3 RE у водоспадуній моделі

У методі водоспаду RE має місце на першому етапі процесу розробки та використовує наступні п’ять видів діяльності: виявлення, аналіз і узгодження, документування, перевірка та управління [31]. При визначенні вимоги консультації із зацікавленими сторонами виявляють вимоги процесу, а при аналізі вимог забезпечується послідовність і повнота виявленої вимоги [7]. Документація, перевірка вимог та управління також здійснюються систематично.

1.2.4 Розробка вимог у методах Agile і Waterfall

Хоча мета RE є однаковою для всіх програмних методів, спосіб RE, що виконується в Agile і Waterfall, є протилежними за своєю природою [4] [31]. У цьому розділі RE в Agile і Waterfall обговорюються з точки зору їх відмінностей щодо залучення клієнтів, нефункціональних вимог, а також використання ітераційного підходу, визначення пріоритетів і документації.

Що стосується залучення клієнтів, RE як в методах Agile, так і в методах водоспаду надають важливе значення залученню зацікавлених сторін, співробітництву обличчям до обличчя та постачанню високоякісних продуктів [18] [31]. Однак, хоча співпраця та комунікація з клієнтами є вирішальними для досягнення високоякісного продукту [25] [18] [7], методи Agile і Waterfall відрізняються в термінах консультацій із клієнтами. У Waterfall клієнти задіяні на початкових етапах збору та аналізу вимог, тоді як у Agile клієнти задіяні протягом усього процесу [31].

Методи Agile і Waterfall також відрізняються щодо використання нефункціональних вимог. У підходах Agile NFR належним чином не ідентифікуються, не моделюються або не зв’язуються на початковій фазі аналізу вимог [14] [15]. Через природу вимог, що змінюються, такі NFR, як ресурси, ремонтопридатність, портативність, безпека або продуктивність, часто ігноруються в Agile [31]. Крім того, команди, які використовують Agile-методи, не часто розглядають NFR як вирішальні чи надзвичайно важливі [34] [13]. Навпаки, існують деякі підходи до боротьби з NFR, які, швидше за все, будуть використовуватися у водоспаді, де моделі та дизайн ретельно розроблені; тоді NFR може бути краще оброблено у Waterfall.

Використання ітераційного підходу в RE в Agile суттєво відрізняється від методу Waterfall. Наприклад, Као і Рамеш [50] виявили, що одна з ключових відмінностей між методами Agile і Waterfall полягає в тому, що перший використовує ітеративний підхід у виявленні вимог для задоволення клієнтів. Замовники отримують робочі функції в більш короткі терміни і беруть безпосередню участь у

зміні вимог [25]. На відміну від цього, дії РЕ виконуються заздалегідь за методом водоспаду і не повторюються протягом усього проекту.

Щодо визначення пріоритетів вимог, Agile і Waterfall використовують різні методи. У Agile використовуються методи, які дозволяють клієнтам надавати інформацію про бізнес-цінність функцій, які відповідають їхнім функціональним вимогам на певній ітерації [25]. Отже, визначення пріоритетів у Agile відбувається як частина процесу ітерації [45]. Для ітерацій вимоги ранжуються та підтримуються як частина списку незавершених історій користувачів. У Waterfall повні вимоги до всього проекту визначаються пріоритетами на основі функцій, функціональних можливостей і нефункціональних пов’язаних елементів наперед, і пріоритетність підтримується протягом життєвого циклу проекту, і зміна пріоритетів є важкою [25].

Нарешті, методи Agile і Waterfall відрізняються з точки зору використання документації. Agile методи більш орієнтовані на код [28], і тому знання вимог не відображаються у вичерпній документації. За словами Чапіна [12], переважна більшість методів Agile забезпечує обслуговування програмного забезпечення без документації. Історії користувачів не стають частиною повної документації системи [25] [18] і використовуються лише для внесення змін у будь-який момент протягом короткого циклу розробки. На відміну від цього, традиційні методи, керовані планом, такі як Waterfall, не вносять таких змін під час циклу розробки [18], а документація використовується для обміну знаннями [4] [31].

1.3 Забезпечення якості програмного забезпечення

Метою будь-якого процесу розробки програмного забезпечення є впровадження якісного продукту без дефектів. Забезпечення якості програмного забезпечення (Software Quality Assurance – SQA) є невід'ємною частиною цього процесу [19] незалежно від використовуваного методу розробки програмного забезпечення [21]. Соммервілль [44] визначає якість програмного забезпечення як процес управління, пов’язаний із забезпеченням невеликої кількості дефектів у програмному забезпеченні та досягнення ним необхідних стандартів

ремонтопридатності, надійності, портативності тощо [44]. Він також відноситься до процесу оцінки програмного забезпечення для забезпечення відповідності вимогам до програмного забезпечення [4] [43]. Для покращення якості програмного забезпечення було запропоновано багато підходів, включаючи повне управління якістю, шість сигм, модель зрілості можливостей тощо [43]. Покращення якості допомагають операційній продуктивності кількома способами, наприклад, збільшення доходу, зниження витрат і підвищення продуктивності [27]. Діяльність SQA включає аудит, тестування програмного забезпечення та контрольну точку поінформованості для перевірки статусу проекту [42]. Згідно з [6], ці дії зазвичай виконуються для визначення придатності або цінності програмного продукту для його операційної місії.

Через природу програмного забезпечення важко оцінити якість, використовуючи однаковий набір рекомендацій для всіх продуктів у всіх галузях [3]. Багато дослідників придумали стандартизовані моделі, які використовують певні параметри, такі як продуктивність, надійність, чутливість, естетичність, довговічність, придатність до експлуатації тощо, щоб визначити якість [43] [3]. Міжнародна організація зі стандартів (ISO) запропонувала стандарт ISO 9001, який застосовується до низки компаній, що займаються розробкою програмного забезпечення [3] [43]. Хоча спочатку якість визначалася як відповідність стандарту та специфікації, це визначення було змінено в 1990-х роках, щоб не відставати від динамічного бізнес-середовища. Стандартом оцінки якості програмного забезпечення було прийнято ISO 9126 [43]. ISO/IEC 15504 (едосконалення процесу програмного забезпечення та визначення можливостей) і відноситься до «основи для оцінки процесів». Цей стандарт розглядає процес, методи та загальне управління якістю, а також пропонує стандарти для проектування, документації, впровадження, підтримки, моніторингу, контролю, поточного перегляду та оцінки діючих стандартів [3]. Для підвищення повторюваності, передбачуваності та керованості програмного забезпечення були запропоновані різні моделі, такі як CMM (Capability Maturity Model), CMMI (Capability Maturity Model Integration), TMM (Testing Maturity Model), TPI (Test Process Improvement) [19].

1.3.1 Забезпечення якості програмного забезпечення в Agile

Забезпечення якості програмного забезпечення (SQA) в Agile – це тестування програмного забезпечення, яке дотримується тих же принципів, що й метод розробки програмного забезпечення Agile. Дві з найбільш значущих характеристик Agile: 1) здатність обробляти нестабільні вимоги та 2) здатність виконувати за більш короткий проміжок часу [32]. SQA в Agile також вимагає швидшого тестування нестабільних вимог. Щоб задовольнити вимоги, в Agile якість програмного забезпечення підтримується шляхом постійного тестування та роботи з програмним забезпеченням, схваленим клієнтом, протягом кожної ітерації циклу розробки. Agile гарантія якості використовується, щоб гарантувати, що програмне забезпечення може реагувати на зміни, як того вимагає клієнт. Оскільки вимоги Agile можуть змінитися в будь-який момент, ці зміни також необхідно перевірити. Хуо та ін. [32] виявили, що забезпечення якості починається на дуже ранніх етапах і триває на всіх етапах (див. рис.1. 2).

На наведеному рисунку помітно, що методи забезпечення якості Agile виконуються одночасно з іншими процесами, такими як визначення вимог, історії користувачів, планування випуску, а також проектування систем і програмного забезпечення. Це також показує, що ці процеси тривають на етапах впровадження та інтеграції.

Для того, щоб SQA-тестування відбулося в коротші терміни на будь-якому етапі, беруть участь усі члени міжфункціональної команди Agile. Усі члени команди, включаючи аналітиків із забезпечення якості, залучені та заохочуються для забезпечення постачання бізнес-цінності, бажаної клієнтом, через часті проміжки часу [1]. Якість Agile вважається результатом таких практик, як ефективна спільна робота та спілкування під час поступового розвитку та ітеративного розвитку. У цьому процесі залучаються всі члени команди, включаючи аналітиків із забезпечення якості, і заохочується забезпечити надання бажаної клієнту бізнес-цінності через часті проміжки часу [1]. Це означає, що процедури забезпечення якості в Agile Methods залежать від командної співпраці [28], яка служить центром забезпечення якості [4] за більш короткий час.

1.3.2 Забезпечення якості програмного забезпечення в Waterfall

Як згадувалося раніше, модель водоспаду ділить життєвий цикл розробки програмного забезпечення на лінійні етапи з чітко визначеними діями для кожного етапу. Діяльність включає 1) визначення вимог 2) проектування системи та програмного забезпечення 3) впровадження та модульне тестування 4) інтеграцію та тестування системи та 5) експлуатацію та обслуговування [44].

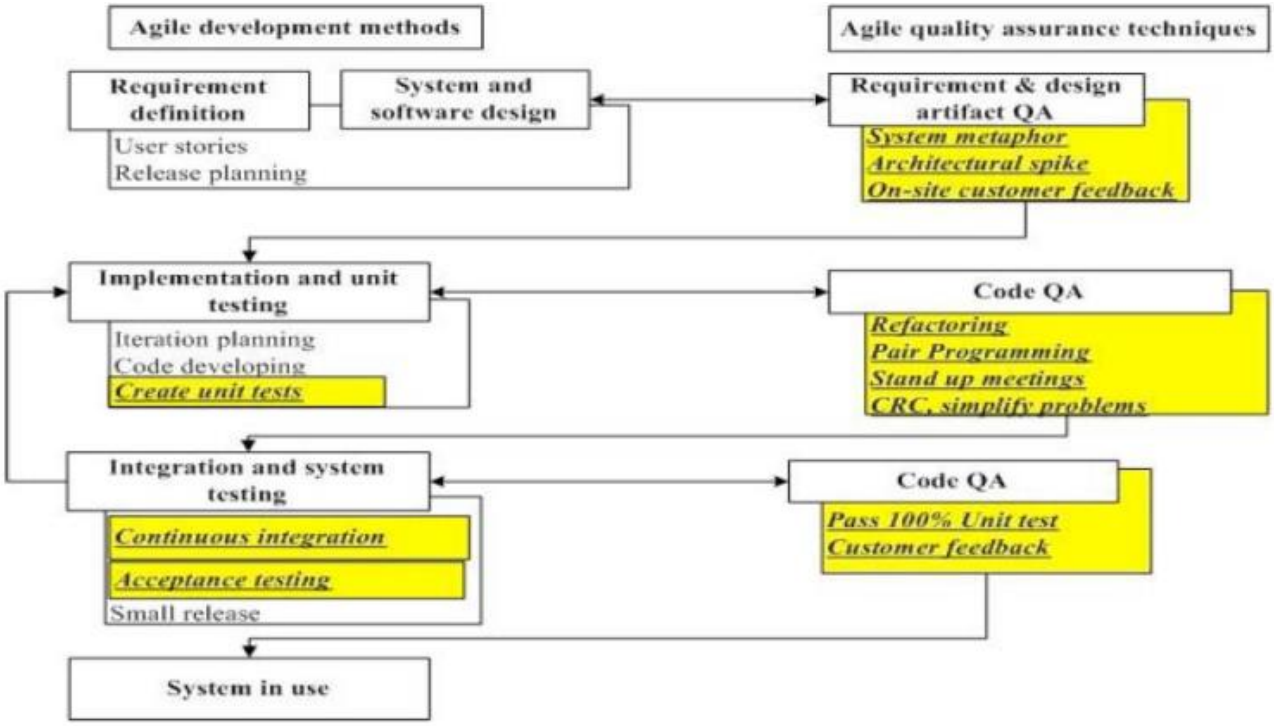

Рисунок 1.2 – Зв'язок між методами Agile розробки та методами забезпечення якості

Оскільки результати однієї діяльності необхідні для наступної діяльності, нові етапи не почнуться, доки не буде завершено попередній етап. У Waterfall традиційний процес перевірки включає в себе дії, які забезпечують створення програмної системи для задоволення замовника за наявності документа специфікації [4], а результати дослідження необхідні як результати. На етапі тестування виконується дослідження того, чи відповідає вимога, надана зацікавленими сторонами, результату продукту.

Етапи тестування методу Waterfall проводяться після завершення розробки [3], і його мета — визначити правильність продукту та здатність рухатися вперед у процесі SQA. Якщо програмне забезпечення не відповідає вимогам і видає помилки, в результаті виникають проблеми та дефекти. Хоча документація проблем і дефектів свідчить про те, що етап тестування завершено, етапи розробки та тестування повторюються, доки всі проблеми та дефекти не будуть усунені. Рисунок 1.3 ілюструє кожен етап і пов'язаний з ним у SQA в моделі водоспаду.

1.3.3 Порівняння якості програмного забезпечення між методом Agile і Waterfall

Використання SQA в методах Agile і Waterfall відрізняється з точки зору початку, частоти та спрямованості діяльності. У Agile дії SQA починаються з самого початку, тоді як у Waterfall дії SQA не починаються, поки не будуть завершені всі розробки. У дослідженні в [30], було відзначено, що здатність виявляти проблеми раніше, щоб забезпечити шанси виправити їх на більш ранньому етапі.

Друга відмінність між методом Agile і Waterfall полягає в тому, як часто виконуються дії SQA. За словами авторів [22], для перевірки вимог у Agile використовуються рання та постійна перевірка та часті зустрічі. Також забезпечення якості відбувається частіше в методі Agile, ніж у Waterfall. Було включено багато показників, таких як проблеми, пов’язані зі спринтом, вихід за межі спринта, кількість перешкод на спринт, а також були введені дошки стану спринтів для вимірювання якості в Agile, щоб врахувати його динамічну природу [1]. Однак існують суперечки щодо гнучкості цих показників, щоб задовольнити мінливі вимоги, а також щодо того, чи є якість клієнтів задовільною чи ні [21].

Фокус діяльності SQA також відрізняється в методах Agile і Waterfall. Основне завдання Agile-тестування полягає в тому, щоб перевірити лише елементи для певної ітерації. На відміну від цього, метод Waterfall концентрується на повних вимогах проекту. У Agile основним акцентом є модульне тестування та приймальне тестування [22]. Загалом, здається, не вистачає значущості для нефункціональних вимог (NFR), крім програмних рішень функціональних вимог [13].

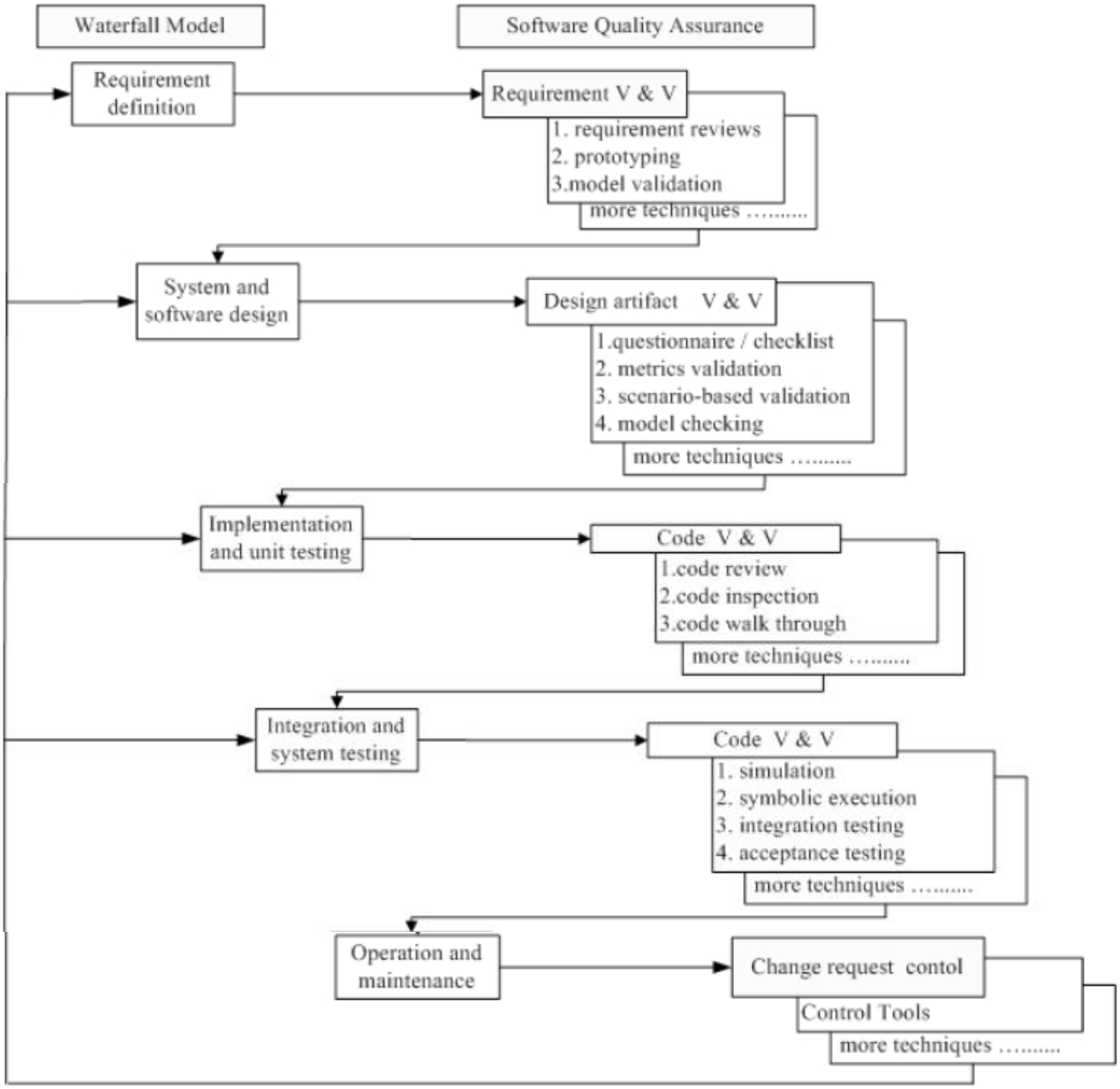

Рисунок 1.3 – Етапи та дії SQA у Waterfall

Це більш очевидно в Agile, ніж у Waterfall. Попередні дослідження з підходами Agile не зосереджувалися на нефункціональних аспектах і припускали, що раннє виявлення нефункціональних вимог може сприяти досягненню покращення якості [48].

2 ПРОПОНОВАНА МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ

Метою розділу є представлення підходу, використаного в цьому емпіричному дослідженні, а також обґрунтування вибору методу дослідження випадку. Спочатку описується обґрунтування вибору підходу до вивчення випадку, а потім опис різних типів методів дослідження випадку та причини їх вибору.

У цьому розділі представлені гіпотези дослідження, джерела даних і критерії відбору випадків, а також кроки, пов’язані зі збором даних.

2.1 Гіпотези дослідницького проекту

Нижче описано дві гіпотези цього проекту:

Гіпотеза 1: кількість дефектів у проектах Agile і Waterfall однакова.

$$\mu_{\text{Agile Кількість дефектів}} = \mu_{\text{Водоспад Кількість дефектів}} \tag{2.1}$$

Альтернативна гіпотеза: кількість дефектів різна в проектах Agile і Waterfall.

$$\mu_{\text{Agile Кількість дефектів}} \neq \mu_{\text{Водоспад Кількість дефектів}} \tag{2.2}$$

де $\mu_{\text{Agile No. of Defects}}$ – середнє сукупність для ряду дефектів на Agile Projects, а $\mu_{\text{Waterfall No. Defects}}$ – середнє сукупності для кількості дефектів на Waterfall Projects.

Гіпотеза 2: кількість дефектів, пов’язаних з NFR, у проектах Agile та Waterfall однакова.

$$\mu_{\text{Agile Кількість дефектів NFR}} = \mu_{\text{Waterfall Кількість дефектів NFR}} \tag{2.3}$$

Альтернативна гіпотеза: кількість дефектів, пов'язаних з NFR, різна в проектах Agile і Waterfall.

$$\mu_{\text{Agile Кількість дефектів NFR}} \neq \mu_{\text{Waterfall Кількість дефектів NFR}}, \tag{2.4}$$

де $\mu_{\text{Agile No. NFR Defects}}$ – середнє сукупність для ряду NFR, пов'язаних з дефектами в Agile Projects, а $\mu_{\text{Waterfall No. NFRDefects}}$ – середнє населення для ряду дефектів, пов'язаних з NFR, у проектах Waterfall.

2.2 Збір даних

Цей розділ описує стратегії вибірки даних. На першому етапі дослідження було чітко визначено мету та формалізовано метод збору даних. Це було вирішальним для розуміння того, які дані потрібно збирати, звідки і чому вони потрібні для перевірки гіпотез.

В якості джерела даних була обрана фінансова організація з середньою командою програмного забезпечення. Ця організація, як правило, пропонує підтримку таким фінансовим установам, як банки та іпотечні кредитори, і має команду програмного забезпечення, яка, як відомо, надає технологічні рішення різному колу клієнтів протягом кількох років за методами Waterfall і Agile. Ця конкретна команда працювала в індустрії, яка швидко розвивається і динамічно розвивається, і тому ця обрана компанія була сприйнята як хороший представник середньої організації з інформаційних технологій.

2.3 Формування групи Agile і Waterfall

Для перевірки гіпотез був необхідний метод для визначення та формування двох окремих груп (наприклад, Agile, Waterfall). Оскільки команда програмного забезпечення прийняла метод Agile у 2012 році, усі проекти, які почалися після 2012 року, були віднесені до групи Agile. Проекти, започатковані до 2012 року, були віднесені до групи «Водоспад». Рік початку проекту використовувався для призначення проекту кожній групі, як показано в таблиці 2.1.

Тридцять проектів, виконаних командою розробників програмного забезпечення, були випадковим чином відібрані з 2010 по 2014 роки, з 15 проектів з

2012-2014 років і 15 проектів з 2010-2011 років, які відповідали методам Agile і Waterfall відповідно.

2.4 Джерела даних для дослідження

Після формування груп необхідно було отримати додаткову інформацію. Для відстеження та зберігання інформації для цієї компанії використовувалася система управління життєвим циклом програми, яка називається Team Foundation Server (TFS). TFS — це продукт, який забезпечує управління вихідним кодом, звітність, управління вимогами та управління проектами для методів Agile і Waterfall. Ця система не тільки зберігала детальну історію всіх проектів, включаючи кількість запитів на зміни (комбінація історій користувачів або робочих елементів і дефектів), але також зберігала витрачені години та кінцевий результат тестування кожного запиту на зміну. Він також відстежував кількість дефектів, створених для кожного запиту на зміну.

Для цілей цього дослідження дані з TFS були доступні та зібрані за допомогою кількох кроків. По-перше, була врахована кількість запитів/змін, щоб зрозуміти розмір проекту. Потім час і зусилля були витрачені на кожен запит, щоб визначити тривалість проекту. У цьому процесі також були записані ім’я та кількість людей, які брали участь у виконанні запитів. Нарешті, також було зібрано кількість дефектів, виявлених під час тестування, та характер проблем. Створивши відповідний запит і звіт від команди

Була створена базова система з такими параметрами: «Кількість запиту на зміну», «Результат перевірки якості» та «Кількість дефектів». На рисунку 5 показано запит, який використовується для збору даних із TFS.

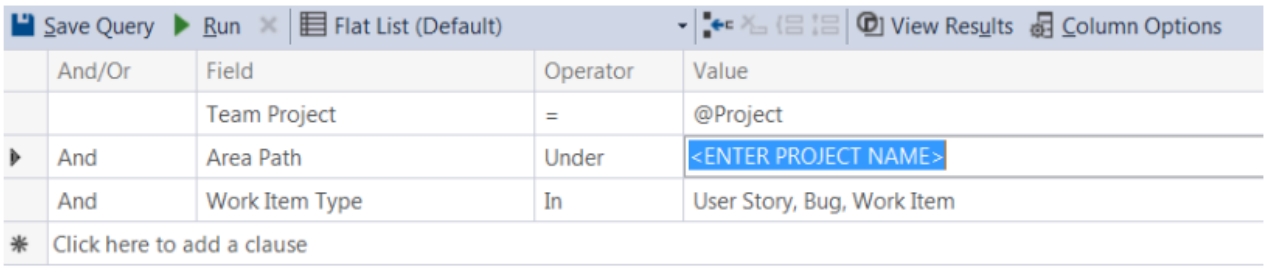

Рисунок 2.1 – Запит, що використовується для виконання звіту з TFS

Мета-характеристика «Кількість запитів на зміну» була використана для визначення розміру проектів та для порівняння, на виконання яких потрібно більше часу. «Кількість дефектів (помилок)» використовувалася для оцінки того, які проекти дали кращу якість, де краща якість пов'язана з меншою кількістю дефектів. У таблиці 2.1 наведено зразок звіту про дефекти з TFS.

Таблиця 2.1 – Зразок звіту про дефекти в проекті

ID	**Тип**	**Витрачені години**	**Назва**	**Знайдено під час**	**Команда Scrum**
36546	Дефект	7.5	<Назва 1>	Розробник/функціональне тестування	<Назва команди>
36616	Дефект	5	<Назва 2>	Етап/внутрішнє приймальне випробування	<Назва команди>
37683	Дефект	3	<Назва 3>	Виробництво/розгортання	<Назва команди>
37688	Дефект	3.25	<Назва 4>	QA/розгортання	<Назва команди>
36545	Дефект	5,75	<Назва 5>	Розробник/функціональне тестування	<Назва команди>
36547	Дефект	4.5	<Назва 6>	Розробник/функціональне тестування	<Назва команди>
36615	Дефект	3.5	<Назва 7>	Етап/внутрішнє приймальне випробування	<Назва команди>
37676	Дефект	1	<Назва 8>	QA/розгортання	<Назва команди>

Вибрані проекти дозволили провести порівняння між методами Agile і Waterfall, щоб зрозуміти загальний вплив. Далі потрібно було вибрати поля даних для

аналізу; також важливо було спочатку зрозуміти характер даних, які були доступні для проекту.

Для відбору кейсів значущими були характеристики проекту та команд. Оскільки метою проекту було порівняння якості методів Agile і Waterfall, було використано три критерії відбору для стандартизації зусиль або зменшення упередженості в кожній із двох груп. Цими критеріями були: розмір проекту, розмір команди та тривалість проекту.

Розмір проекту був найважливішим фактором при виборі проектів для включення в аналіз. Розмір проекту визначався з використанням кількості робочих елементів або кількості історій користувачів для кожного проекту. Робочі елементи та історії користувачів представляють кількість функцій/функцій, які потрібні проекту протягом усього проекту. Нижче наведено детальне пояснення історій користувачів і робочих елементів.

Кожен проект за методом Agile перевірявся за кількістю запитів, які називаються історіями користувачів, у яких вимоги до програмного забезпечення документуються менш формальним способом як основний блок [4], [30]. Розповідь користувача структурована в короткому форматі, який описує природною мовою «Чому» та «Як» проекту [17]. Історії користувачів зазвичай пишуться у формі «Як (роль), я хочу (щось), щоб я міг (користати)». Щоб доповнити неформальну заяву про історію користувача, також включені критерії прийняття, щоб якість програмного забезпечення вважалася виконаною. «Критерії прийнятності» записуються у форматі «Дано», «Коли» та «Тоді» (табл. 2.2).

Таблиця 2.2 – Зразок формату історії користувача та критерії прийняття

Назва	**Схвалити новий запит користувача**
Історія користувача	Як роль «Адміністратор користувача», я хочу затвердити запит користувача, який очікує на розгляд, щоб нові користувачі могли отримати доступ до системи.

Прийняття Критерії	Враховуючи, що я хочу отримати доступ до запиту користувача, який очікує на розгляд Коли я вхожу в систему і відкриваю «Очікує Запити користувачів.' Потім я переглядаю список користувачів із опціями «Затвердити» та «Відхилити». Враховуючи, що я хочу схвалити запит користувача, який очікує на розгляд Коли я знаходжуся на екрані "Запити користувачів, які очікують на розгляд", і Я натискаю на посилання «Затвердити». Потім я переглядаю повідомлення з підтвердженням, яке показує «Користувачу надано доступ до входу».

У проектах, виконаних за методом водоспаду, був представлений традиційний формат, який включав Документ специфікації бізнес-вимог (BRSD). BRSD – це письмовий план, який було завершено до початку фази проектування та впровадження програмного забезпечення. Вимоги в специфікаційних документах розбиті на запити на зміни, які називаються «робочими елементами», які в основному засновані на функціональності системних функцій. На відміну від історій користувачів, робочі елементи не мають узгодженого формату, але вони розбиті за функціями та функціональними можливостями. У деяких випадках один робочий елемент можна перевести на кілька історій користувача, тоді як в інших випадках одну історію користувача можна розбити на кілька робочих елементів. Зрештою, запити у формі історії користувача представляють, як функцію потрібно буде використовувати з точки зору користувача, а робочі елементи представляють, як функція буде використовуватися загалом.

Після збору історій користувачів і робочих елементів для проекту, графіки дисперсії були використані для вибору випадків для аналізу проектів у групі Agile і Waterfall (Рисунок 2.2 і Рисунок 2.3).

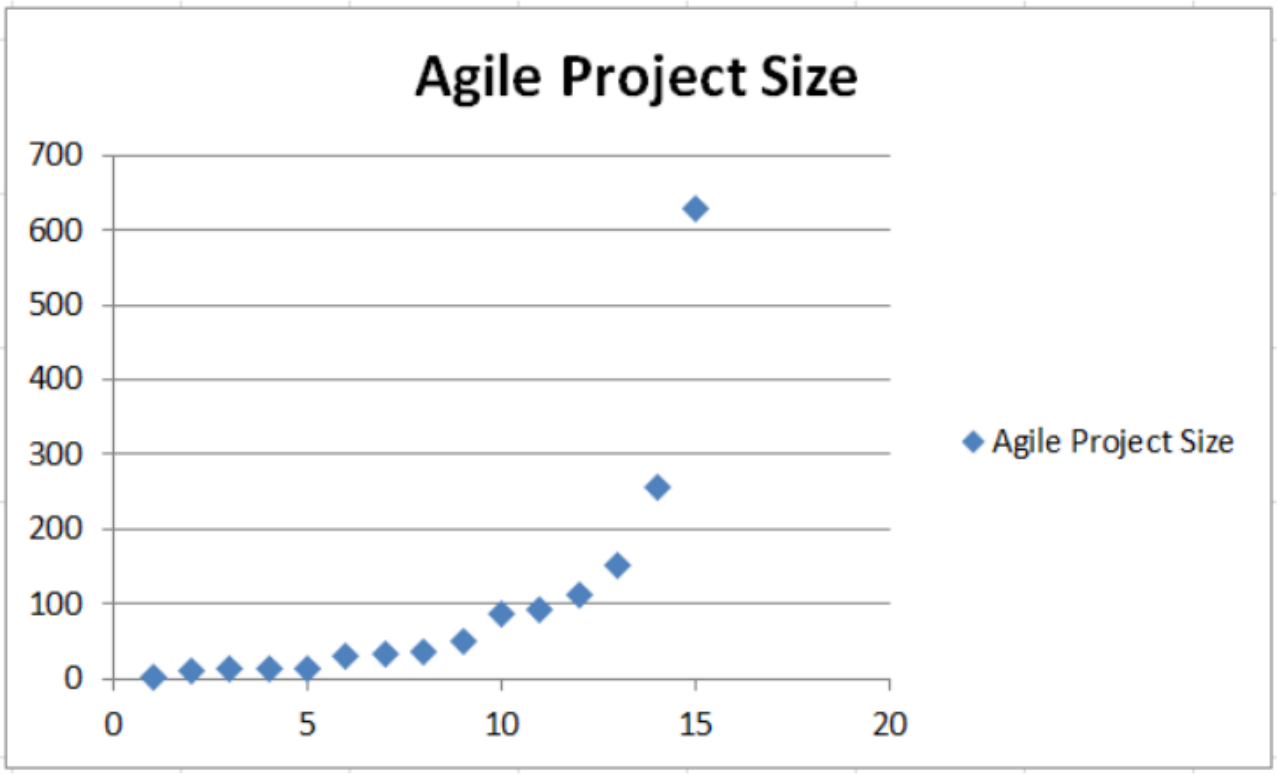

Рисунок 2.2 – Розмір вибраних проектів за допомогою методу Agile

По-друге, було визначено, що розмір команди може бути фактором, який може порушити порівняння між методами Agile і Waterfall, і, отже, міри дисперсії цієї змінної також були враховані при виборі випадків. Під час перевірки було встановлено, що команда програмного забезпечення складалася з 35-40 членів, які були розбиті на менші підгрупи для виконання різних проектів. Наприклад, для проекту середнього розміру команда зазвичай складається з менеджера проекту, власника продукту, 2-6 розробників і 1 аналітика із забезпечення якості.

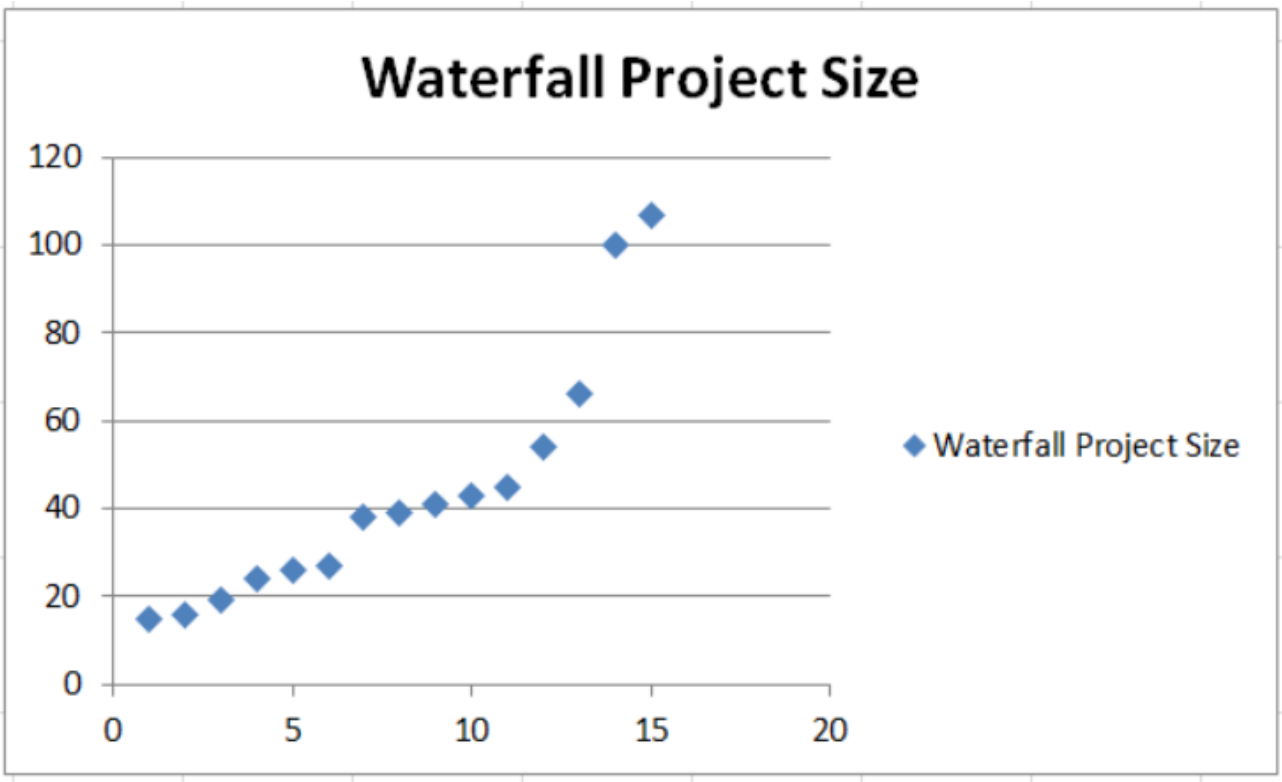

Рисунок 2.3 – Розмір вибраних проектів за допомогою методу Agile

Було помічено, що деякі проекти мали невеликі команди, що складалися з 2-4 членів, тоді як інші проекти мали помірний розмір команди (5-11 членів), а решта мали велику команду (12-18 членів). Таким чином, мета полягала в тому, щоб вибрати випадки, які мали команди схожого розміру, враховуючи, що успіх методу Agile сильно залежить від командної співпраці та спілкування. Помірний розмір команди мав пріоритет для відбору проектів.

Нарешті, тривалість проекту вважалася важливим фактором при виборі випадку. Більш тривала тривалість проекту невеликого розміру свідчить про те, що на осмислення та виконання завдань було дозволено більше часу порівняно з проектом, виконаним за коротші терміни. Тривалість розраховувалась шляхом вивчення часу, який пройшов від початку проекту до його завершення. У межах 15 випадково відібраних проектів у групі Agile та Waterfall тривалість становила від 1 місяця до 2 років. Проекти, які мали тривалість 3-9 місяців, були визнані придатними для включення, а випадки з меншою тривалістю (1-2 місяці) і більш тривалими (10-

24 місяці) були виключені. Рисунки 2.4 і 2.5 ілюструють подібну тривалість проектів у групах Agile і Waterfall.

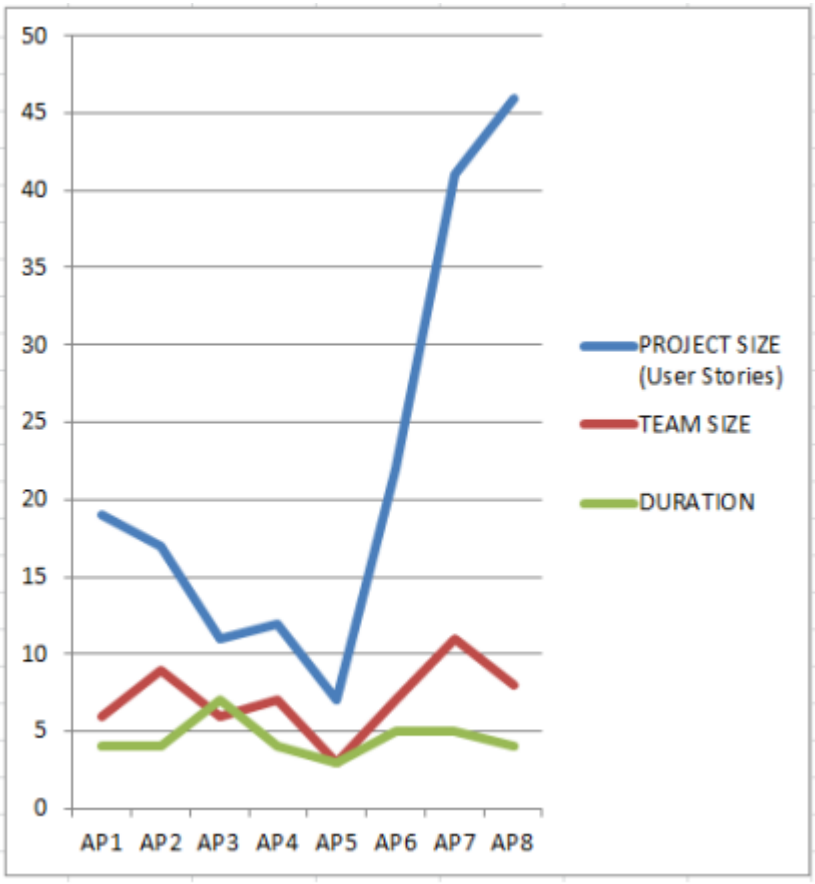

Рисунок 2.4 – Розмір команди, тривалість проекту та розмір проекту для кожного Agile-проекту, вибраного для аналізу

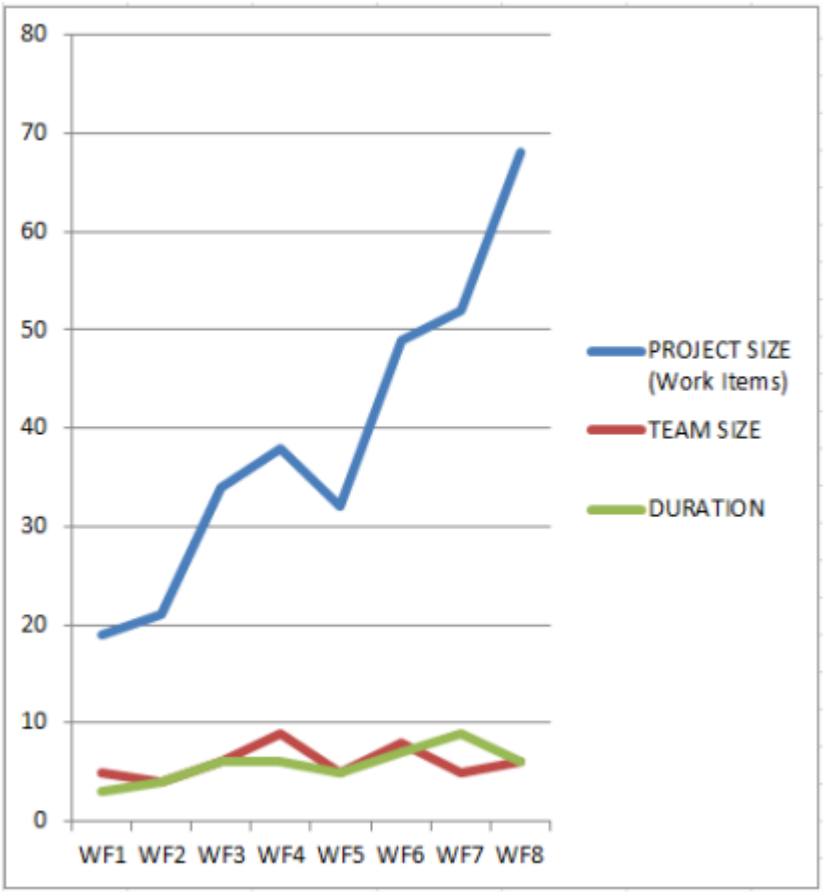

Рисунок 2.5 – Розмір команди, тривалість проекту та розмір проекту для кожного проекту водоспаду, вибраного для аналізу

Примітно також, що розмір команди та тривалість проектів узгоджуються, а розмір проекту відзначається більшою мінливістю.

2.5 Опис Agile та Waterfall проектів

У таблицях 5 і 6 описано вибрані проекти Agile і Waterfall для цього дослідження.

Таблиця 2.3 – Опис Agile-проектів

Немає.	**Опис**
AP1	Для цього проекту команда з 6 осіб виконала 19 запитів за 4 місяці, щоб клієнт міг отримати доступ і відстежувати свій статус за допомогою сповіщень електронною поштою.
AP2	Цей проект був започаткований, щоб дозволити співробітникам самостійно зареєструватися на веб-порталі, а також отримати доступ до іпотечних документів для своїх клієнтів. Це дозволило керівникам схвалювати співробітників, які запитали доступ до веб-порталу. У цьому проекті брали участь 9 людей (включаючи 1 менеджера проекту, 1 власника продукту, 1 аналітика із забезпечення якості, 1 керівника групи розробників, 4 розробників та 1 аналітика з питань прийняття користувачів) і тривав 4 місяці. Протягом процесу він був розбитий на 17 елементів.
AP3	Метою цього проекту було вдосконалення конкретного процесу фінансової компанії, включаючи покращення відстеження угод та автоматизацію звітності. Цей проект стартував у 2013 році і тривав 7 місяців. Шість членів команди (включаючи 1 менеджера проекту, 1 власника продукту, 1 аналітика із забезпечення якості, 1 керівника групи розробників, 1 розробника та 1 аналітика прийняття користувачів) були залучені до проекту для розміру 11 історій користувачів.
AP4	Для цього проекту були внесені зміни в існуюче програмне забезпечення, що дозволило подавати кілька замовлень і скасовувати замовлення. Всього за 4 місяці було завершено деякі роботи з обслуговування (розбитих на 12 історій користувачів) існуючої програми.
AP5	Цей проект був невеликою частиною великого проекту, який мав багато результатів. У рамках цього підпроекту було проаналізовано та заплановано лише кілька пунктів, які мають справу з інтеграцією сторонніх додатків. Проект було завершено понад 3 місяці та розпочато наприкінці 2013 року. П’ять членів команди (1 власник продукту, 1 розробник і 1 аналітик із забезпечення якості) завершили проект розміром 7.
AP6	Цей проект був ініційований за запитом від існуючого клієнта. Це передбачало модифікацію існуючої веб-програми, щоб включити нову роль з новими функціями та функціональними можливостями. Команда складалася з 7 членів (1 менеджер проекту, 1 власник продукту, 3 розробники та 1 аналітик із забезпечення якості), а 22 історії користувачів було завершено за 5 місяців.
AP7	Цей проект передбачав запуск нового продукту для клієнта. Основна увага полягала в налагодженні зв'язку між системами та розробці/вдосконаленні системних рішень, які обробляють і відстежують операції, пов'язані з іпотекою, на веб-порталі. Цей проект був завершений, починаючи з 2013 року, протягом 5 місяців і залучив 11 членів команди для 41 історії користувачів.
AP8	Це був найбільший проект у групі з 46 запитами та виконаний командою з 8 осіб протягом 4 місяців. Це дозволило установі іпотечного кредитування отримати дані та фактори ризику з внутрішньої системи обробки.

Таблиця 2.4 – Опис проектів водоспаду

Немає.	**Опис**
WF1	Цей проект був завершений для створення маркетингового порталу для відправки та отримання маркетингових комунікацій на існуючому веб-порталі. Команда з 5 осіб завершила цей проект (19 робіт) протягом 3 місяців.
WF2	Цей проект мав надати прикладні функції, які пропонували економічно ефективні програми для поточної обробки іпотеки та управління оцінкою. Він був завершений у 2010 році за 4 місяці командою з 4 осіб, включаючи 1 бізнес-аналітика, 2 розробників та 1 аналітика із забезпечення якості із 21 робочою позицією.
WF3	Цей проект передбачав впровадження та впровадження нового веб-порталу, де існуючі та нові клієнти могли подавати інструкції та відстежувати хід файлів, щоб забезпечити послідовність та ефективність. Цей проект було завершено з 34 предметами за 6 місяців. Команда з 6 осіб, включаючи 1 менеджера проекту, 1 бізнес-аналітика, 3 розробників та 1 аналітика із забезпечення якості, завершила проект.
WF4	Цей проект передбачав розробку веб-додатку, який дозволяв подавати, відстежувати та звітувати про операції з оцінки нерухомості через онлайн-портал, а також можливість відстежувати операції «рефінансування». Для цього проекту була задіяна команда з 9 осіб, у тому числі 1 бізнес-аналітик, 6 розробників та 1 аналітик із забезпечення якості, щоб вирішити 38 питань протягом одного місяця.
WF5	Цей проект покращив існуючі функції програми, щоб надати можливість керувати інструкціями, фінансуванням та звітністю щодо запитів на покупку та рефінансування фінансової установи. У цьому проекті була команда з 5 людей, що складалися з 1 бізнес-аналітика, 2 розробників та 1 аналітика із забезпечення якості, які працювали протягом 5 місяців, щоб виконати 32 завдання.
WF6	Цей проект передбачав створення програми, яка дозволяла клієнтам розміщувати замовлення на послуги оцінки, переглядати/відстежувати статус оцінки та отримувати результати процесу оцінки через веб-портал. Цей проект був завершений протягом 7 місяців із 8 членами команди, яка складалася з 1 керівника проекту, 1 бізнес-аналітика, 4 розробників та 2 аналітиків із забезпечення якості. Розмір цього проекту включав 49 елементів, що стало найбільшим із шести відібраних проектів водоспаду.
WF7	Цей проект був започаткований з метою збільшення економії коштів шляхом оптимізації процесу іпотечних операцій для клієнта. У центрі уваги проекту було скорочення кількості ручних кроків, пов’язаних з остаточною звітністю та узгодженням файлів, за рахунок скорочення друку та доставки документів. Розмір цього проекту був другим за величиною з шести випадків водоспаду, відібраних для аналізу (52 елементи). Для завершення проекту за 9 місяців було залучено п’ять осіб.
WF8	Проект WF8 був найбільшим із 68 запитами, виконаними командою з 6 осіб протягом 6 місяців. Метою цього проекту була інтеграція зовнішнього бухгалтерського програмного забезпечення з внутрішнім програмним забезпеченням.

Щоб проаналізувати дані та перевірити гіпотези, були обрані наступні характеристики (змінні) результату на основі конкретних інтересів цього дослідницького проекту.

Дефекти.

Різниця між очікуваними та фактичними результатами на етапі забезпечення якості програмного забезпечення відома як дефекти. Різні організації мають різні назви для опису цієї варіації, але зазвичай використовувані терміни для опису дефектів включають помилки, проблеми, інциденти або проблеми. Усі розглянуті дефекти були виявлені після розгортання програмного забезпечення, а не під час розробки програмного забезпечення

Кількість дефектів.

Це кількість дефектів, які були виявлені під час етапу забезпечення якості проекту і є показником якості проекту. Більше недоліків означало, що було виявлено більше проблем, які потрібно було виправити, перш ніж проект можна було вважати завершеним. Для цілей цього дослідження розглядалися лише дефекти, пов’язані з вимогами, незалежно від їх складності. Потім ці дефекти були розділені на 2 групи: дефекти, пов’язані з нефункціональними вимогами (NFR) і дефекти, пов’язані з функціональними вимогами (FR).

Кількість дефектів, пов'язаних з NFR.

Дефекти, віднесені до категорії дефектів, пов’язаних з NFR, були засновані на їх характеристиках.

Кількість пов'язаних з FR дефектів.

Як зазначалося вище, дефекти, пов’язані з FR, також були класифіковані на основі природи дефектів.

Загрози для достовірності длослідження

У цьому розділі зосереджено на загальній достовірності дослідження тематичного дослідження та на тому, чи пов’язані ефекти, які спостерігаються в цьому дослідженні, маніпулюванням незалежною змінною, а не якимось іншим фактором.

Загрози внутрішні.

Цей розділ зосереджується на загальній валідності дослідження тематичного дослідження. Упереджений вибір суб’єктів , як загроза внутрішній валідності, контролювався шляхом ретельного відбору проектів за критеріями відбору: розмір проекту, розмір команди та тривалість. *Історія* як загроза внутрішній валідності була

мінімізована шляхом відбору проектів останніх років. Інструменти, як такі, що мають проекти, контролюють загрозу внутрішній валідності, що проводиться тією ж командою та в подібних умовах. Інші загрози внутрішній валідності, такі як експериментальна смертність, статистична регресія, дозрівання та тестування, не застосовуються до цього дослідження.

Загрози зовнішні.

Взаємодія між упередженнями відбору та незалежною змінною контролювалися за рахунок початкового випадкового відбору вибірки. Реактивне тестування, реактивні ефекти експериментальних домовленостей та загрози багаторазового втручання не застосовуються до цього дослідження, оскільки не передбачається попереднє тестування, яке могло б вплинути на реакції на експериментальні змінні.

Зовнішні загрози стосуються можливості узагальнення результатів за межами експерименту. По-перше, яка б була різниця, якби цей експеримент проводився з іншими проектами? Далі, які були б результати, якби вони були зроблені для різних галузей? Крім того, якими будуть результати для набагато більшого розміру вибірки? Щоб переконатися, що цей проект дослідження є дійсним, типи дефектів будь-якої тяжкості, які розглядалися для проекту, були ретельно розглянуті. Були враховані лише дефекти, пов’язані з вимогами, і про них повідомлялося в кінці під час приймального тестування користувачами.

3 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Метою аналізу було визначити, чи відрізняється якість проектів між методами Agile та Waterfall. У цьому розділі описано емпіричні результати процесу збору та аналізу даних. Спочатку було проведено внутрішньогруповий аналіз, щоб оцінити розмір проекту та зусилля в кожній із двох груп. Потім у цій роботі було перевірено, чи відрізняються методи програмного забезпечення Waterfall та Agile щодо якості, використовуючи кількість дефектів як міру якості.

Мотивація зосередити це дослідження на кількості дефектів була викликана суперечливою поведінкою щодо документації під час порівняння методів Agile і Waterfall. Іти в ногу з сучасними швидко мінливими потребами клієнтів; Гнучкі методи сприяють швидшому виконанню проекту за короткі терміни та дозволяють вносити зміни в будь-який момент. Однак у цьому процесі забезпечення якості програмного забезпечення стає більш складним без знання та розуміння вимог. Оскільки Agile має майже нульову документацію, дані, пов’язані з тривалістю діяльності та бюджетними наслідками, вважалося важко, якщо взагалі неможливим, для порівняння, і їх слід розглядати окремо. Таким чином, порівняння якості з точки зору тривалості проекту та бюджетних наслідків було залишено для подальшої роботи. Нарешті, після обговорення результатів з точки зору інших досліджень, які проводили подібні порівняння, представлені переваги та обмеження цього проекту.

3.1 Аналіз історій користувачів і дефектів у групі Agile

Середнє значення частки історій користувачів усіх Agile-проектів становило 21,9, а середнє значення частки дефектів – 12,5. Рисунок 3.1 ілюструє частку історій та дефектів користувачів для кожного з проектів Agile. Дефекти становлять прибл. 30%-60% порівняно з історіями користувачів у групі Agile.

Подальший аналіз був проведений щодо дефектів, пов’язаних з нефункціональними та функціональними вимогами. У групі Agile середня кількість

дефектів, пов’язаних з NFR, становила 8,6 (SD 3,8), тоді як 3,9 (SD 1,7) дефектів були пов’язані з FR.

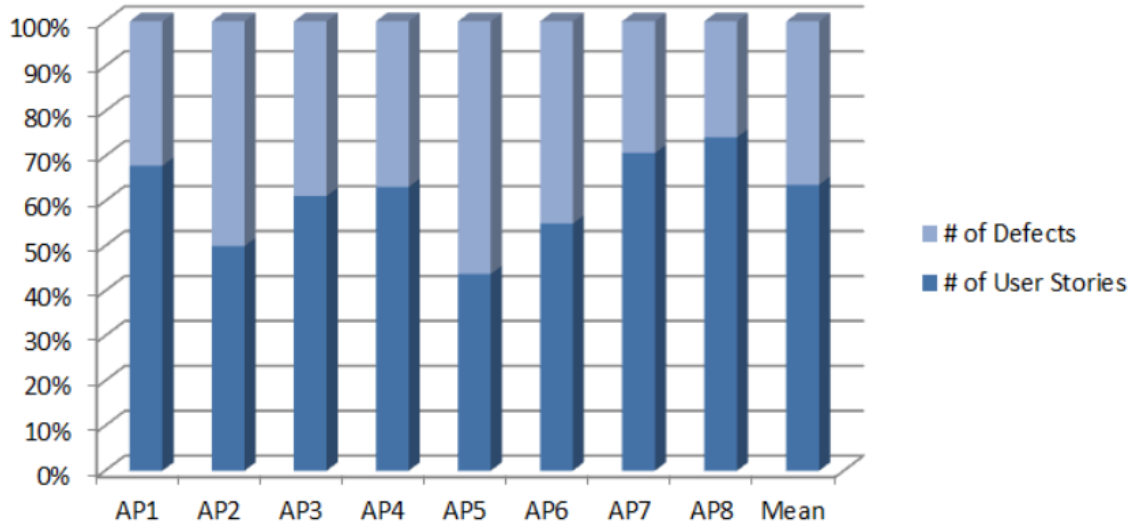

Рисунок 3.1 – Agile проекти: частка історій користувачів та дефектів

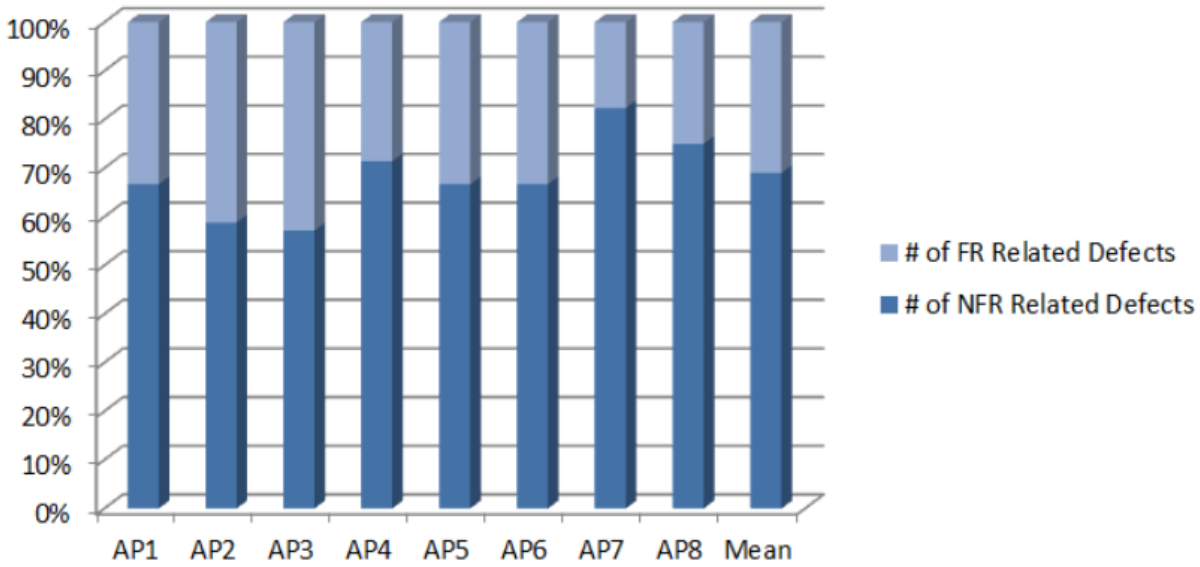

Рисунок 3.2 – Agile проекти: дефекти з часткою дефектів, пов'язаних з NFR і FR

Рисунок 3.2 ілюструє частку дефектів NFR і FR у кожному Agile-проекті. Це показує, що понад 60% були дефектами, пов’язаними з NFR, у цій групі.

3.2 Аналіз робочих елементів і дефектів у водоспадних проектах

Середнє значення всіх робочих елементів у проектах Waterfall становило 39,1 (SD 16,5) робочих елементів і 8 (SD 4,2) дефектів. На рисунку 3.3 показана частка дефектів у кожному з проектів Waterfall, яка впала в діапазоні 8-33% порівняно з історіями користувачів і робочими елементами.

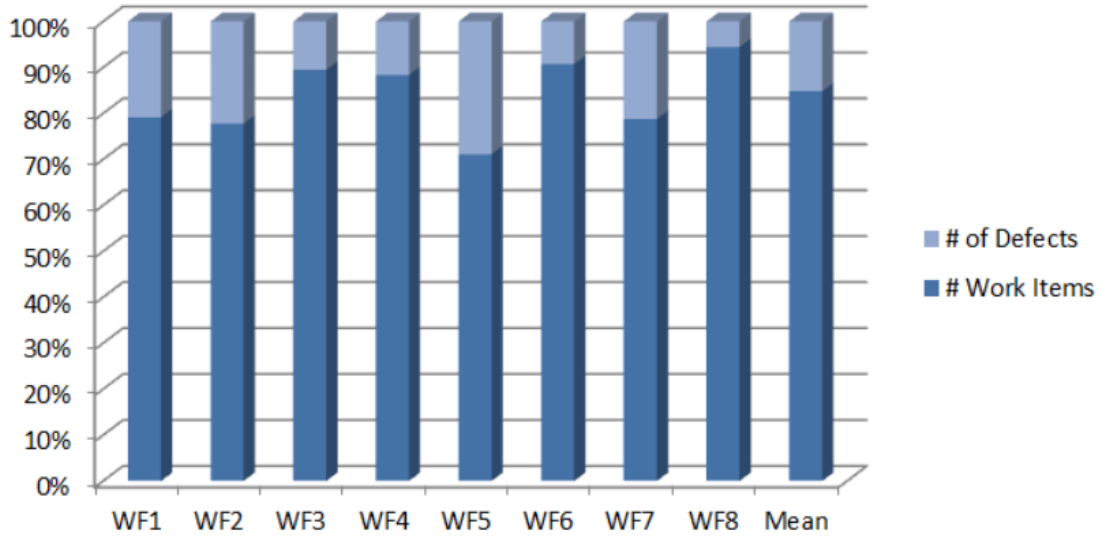

Рисунок 3.3 – Водоспадні проекти: частка робочих елементів і дефектів

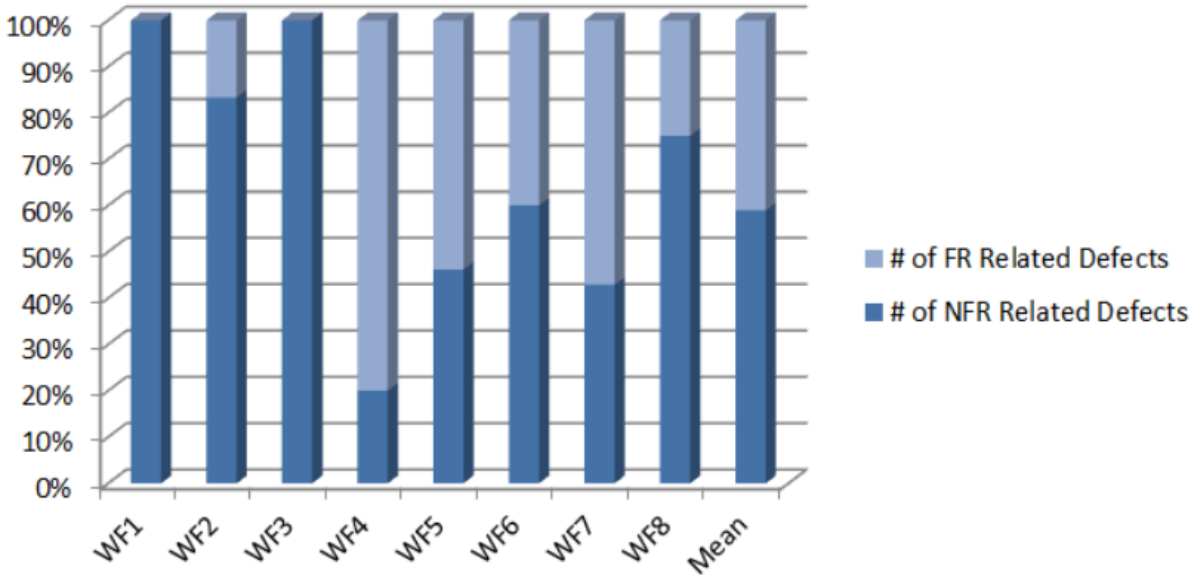

Рисунок 3.4 – Водоспадні проекти: частка дефектів, пов'язаних з NFR і FR

Далі, коли дефекти були проаналізовані та згруповані в категорії, пов’язані з нефункціональними та функціональними вимогами, у групі Waterfall середня кількість дефектів, пов’язаних з NFR, становила 4,1 (SD 1,7), тоді як 3,9 (SD 3,7) дефектів були пов’язаними з FR. . Рисунок 3.4 ілюструє частку дефектів NFR і FR у кожному проекті водоспаду. Ця цифра показує, що кілька проектів не мали жодних дефектів, пов’язаних з NFR, але в середньому понад 50% дефектів складалися з дефектів, пов’язаних з NFR.

3.3 Порівняння між Agile та Waterfall проектами

У цьому розділі порівнюються проекти з групи Agile і Waterfall за кількістю дефектів і розглядаються гіпотези. Таблиця нижче показує групову статистику груп Agile і Waterfall.

Таблиця 3.1 – Групова статистика Agile і Waterfall

	Група	**Середнє**	**Std. Відхилення**	**Std. Помилка Середня**
Розмір проекту	Agile	21.9	14.2	5.0
	водоспад	39.1	16.5	5.8
Кількість дефектів	Agile	12.5	4.9	1.7
	водоспад	8.0	4.2	1.5
Кількість дефектів NFR	Agile	8.6	3.8	1.3
	водоспад	4.1	1.7	0.6
Кількість дефектів FR	Agile	3.9	1.7	0.6
	водоспад	3.9	3.7	1.3

3.3.1 Перевірка гіпотези 1

Була значна різниця в кількості дефектів для Agile (M=12,5, SD=4,9) і Waterfall (M=8, SD=4,2). Проекти з використанням методу Agile мали значно більше дефектів, ніж проекти, що використовували метод Waterfall. У цьому випадку була прийнята альтернативна гіпотеза, і результати показали, що проекти в Agile викликали більшу

кількість дефектів. На рисунку нижче показано обидва значення числа дефектів для Agile і Waterfall.

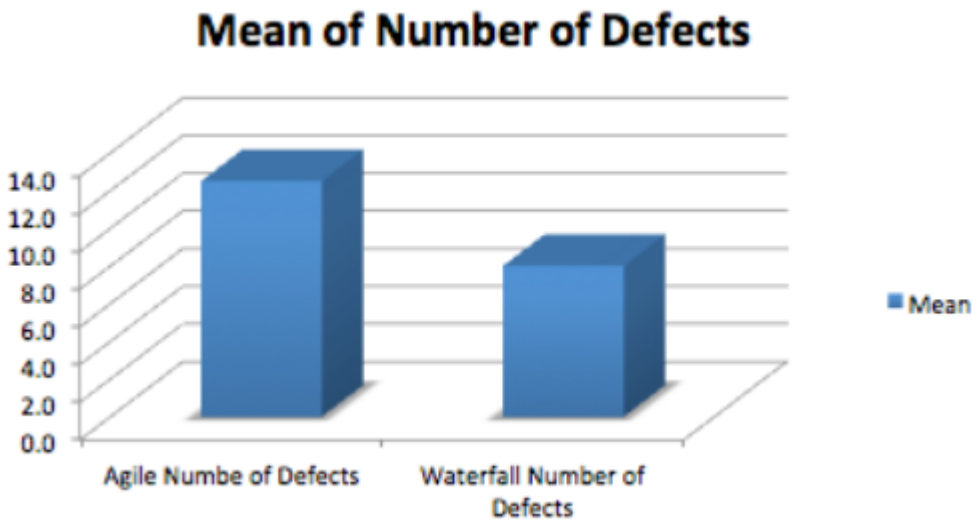

Рисунок 3.5 – Середнє число дефектів для Agile і Waterfall

На сьогоднішній день вдалося знайти лише одне попереднє дослідження, яке показало, що метод Waterfall забезпечував кращу якість, ніж метод Agile. Standish Group, яка з 1985 року збирала інформацію про реальні IT-середовища та проекти розробки програмного забезпечення, перевірила успіхи та невдачі проектів у понад 90 000 IT-проектів, повідомила, що з проектів, завершених у період з 2003 по 2012 рік, 48% проектів Agile проекти були «проблемними» порівняно з 43% проектів водоспаду. Проекти, що розглядались, були визначені як проекти, які були виконані із запізненням, понад бюджет та/або з меншим, ніж необхідно, функціоналом [46].

Навпаки, деякі дослідження показали, що методи Agile мають переваги перед методом водоспаду. Наприклад, у аналізі [32] оцінили відмінності в якості між методами Waterfall і Agile в умовах тиску часу та нестабільного середовища вимог. Вони дійшли висновку, що можливості SQA, частота та час впровадження можуть сприяти успіху методу Agile за цих умов. Вони також визнають, що порівняти якість, отриману в результаті використання Waterfall і Agile, важко через різницю в початкових умовах розробки та витратах [32].

В іншому дослідженні [26], яке порівнювало два випуски одного і того ж продукту, виконані одним і тим же персоналом, виявилося, що 65% покращення якості перед випуском і 35% покращення якості після випуску було відзначено в методах, що використовують XP, порівняно з Waterfall. . Хоча кількість дефектів використовувалася як міра якості, розмір нового випуску становив одну п’яту від старого випуску, який порівнювали з цим дослідженням. Значно менший випуск, можливо, сприяв покращенню якості XP у порівнянні з Waterfall. Таким чином, результати цього дослідження свідчать про те, що при використанні більш інклюзивної та менш упередженої вибірки Agile забезпечить проекти зі значно більшою кількістю дефектів, ніж водоспад.

3.4 Перевірка гіпотези 2

Кількість дефектів, пов’язаних з NFR для Agile (M=8,6, SD=3,8), була значно вищою, ніж у Waterfall (M=4,1, SD=1,7). Agile мав більше дефектів, пов’язаних з NFR, ніж метод Waterfall, що припускало, що нефункціональні вимоги можуть бути не так чітко зрозумілі в Agile.

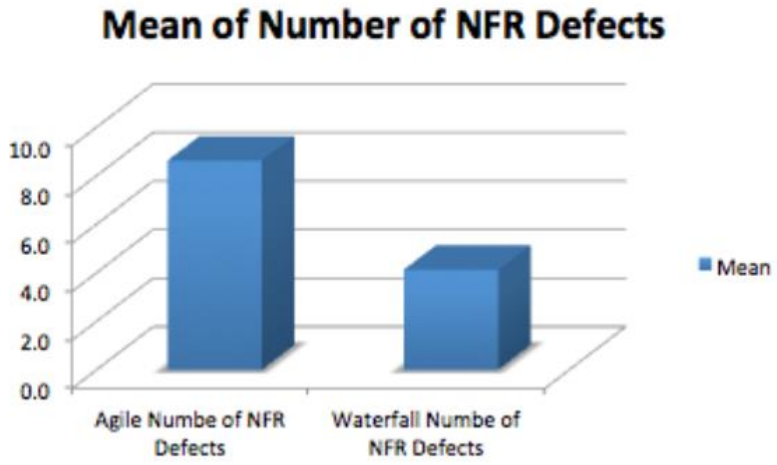

Рисунок 3.6 – Середнє значення кількості дефектів, пов'язаних з NFR, для Agile і Waterfall

Тому була обрана альтернативна гіпотеза. Насправді, з огляду літератури не вдалося виявити жодного суттєвого підходу, де б чітко розглядалися NFR. На рисунку

нижче показано обидва середні числа для дефектів, пов’язаних з NFR, для Agile і Waterfall.

Багато робіт підтверджують, що метод Agile менше справляється з NFR в порівнянні з Waterfall [6][13][14][15][48]. Однією із запропонованих причин цього є те, що NFR не завжди є очевидними при роботі з вимогами в спринтах, які мають функціональність як фокус. Хоча серія ISO 9126 надає показники вимірювання якості та різні рекомендації щодо оцінки якості для загального програмного проекту, ці підходи базувалися на чітко визначених документах, яких немає в Agile [48]. Крім того, Agile не змоделював належним чином нефункціональні вимоги (NFR) та їх потенційні рішення (операціоналізацію) на ранніх етапах розробки [14].

Хоча не було жодних емпіричних досліджень, які б безпосередньо порівнювали дефекти NFR в різних методах розробки програмного забезпечення, в [24] вказано, що 68% проектів Agile повідомляли про використання NFR під час оцінки порівняно з 40% для Waterfall. Ці дані свідчать про те, що метод Agile включає більше NFR, ніж можна було б очікувати, якщо порівнювати з методом Waterfall.

Результати цієї роботи підтверджують необхідність постійної уваги до розуміння, документування та передачі вимог у процесі розробки програмного забезпечення. Як згадувалося в попередніх розділах, документація вичерпних і повних вимог присутня на початку проекту Waterfall, але в Agile вимоги повідомляються поетапно і менш формальним способом. Незважаючи на те, що методи Agile набули популярності, багато практиків і дослідників не знають, як Agile виконує вимоги [31], і сумніваються в його перевагах [21].

Результати цього дослідження показують, що метод, який використовував повні та стабільні вимоги на початку проекту (тобто, Waterfall), викликав менше дефектів, пов’язаних із вимогами, і, таким чином, підтримує цінність розуміння вимог на початку процесу розробки програмного забезпечення.

3.5 Сильні сторони та обмеження результатів дослідження

Хоча існують загальні дослідження щодо розробки вимог (RE) та забезпечення якості програмного забезпечення (SQA) Agile, є лише кілька емпіричних досліджень, щоб зрозуміти вплив RE на SQA в Agile. Таким чином, ці висновки сприяють впливу RE на SQA в Agile.

По-перше, одна з сильних сторін цієї роботи полягає в її дослідженні матеріалів і реального світу (тобто екологічна обґрунтованість). Це має практичне застосування, яке є актуальним і корисним для життя. Цей тип дослідження важко повторити в контрольованому середовищі з обмеженими ресурсами. Дані свідчать про те, що процес впровадження Agile повинен бути добре продуманий таким чином, щоб приділяти більше уваги вимогам.

По-друге, це дослідження охопило поєднання проектів, які обслуговували різних клієнтів у фінансовій галузі. Хоча є дослідження в інших секторах, жодне не проводилося в подібній організації. Крім того, наявність різних клієнтів для кожного проекту дає можливість отримати зворотній зв’язок з кількох джерел.

Нарешті, оскільки результати цієї роботи свідчать про те, що проблеми, пов’язані з NFR, були відповідальними за те, що Agile представляв більшу кількість дефектів, ніж Waterfall, компанії, які прийняли або планують застосувати метод Agile, повинні розглядати NFR як обов’язкову частину RE. Інакше заощадження завдяки прискоренню процесу розробки та меншому бюджету можуть бути скасовані після розгортання в результаті більше часу та грошей, витрачених на усунення дефектів, яких можна було б уникнути.

Метою цього дослідницького проекту було емпірично вивчити, чи забезпечує метод Agile продукт кращої якості. Метод кейсового дослідження вибрано таким чином, щоб інформація могла бути корисною в подібних умовах, щоб краще зрозуміти, який із двох методів, Agile і Waterfall, можна застосувати. У цьому розділі обговорюються обмеження, пов’язані з дослідженням конкретного випадку.

Перше обмеження пов’язане з вибором проектів. Як це може статися в більшості ситуацій, коли таке дослідження може бути проведено, розробники не були

однаковими для всіх проектів. Однак ці розробники належали до менших підкоманд, які були частиною однієї великої команди під одним керівництвом. Цей факт може пом’якшити будь-які міркування щодо упередженості вибору.

Далі, збір даних щодо кількості дефектів може мати невелике упередження до методу Agile. Навряд чи в процесі водоспаду, під час Agile-проектів деякі дефекти можуть навіть не реєструватися як такі. Вони могли бути виявлені під час взаємодії для усунення іншого дефекту, і оскільки Agile має політику документування якомога менше, ніяких анотацій щодо цього нового дефекту не було б зроблено. Однак, спостерігаючи та бравши участь у подібних проектах в тій самій організації (не використано в цьому дослідженні), можна стверджувати, що кількість дефектів, які могли бути не зареєстровані, була б недостатньою, щоб змінити результати.

Насправді це могло лише закріпити результати, отримані в цій роботі.

У цьому розділі обговорюються основні висновки цієї роботи та результати інших подібних досліджень. Порівнюючи результати подібних досліджень та переглядаючи сильні та слабкі сторони, було очевидно, що вплив відсутності вимог та неадекватного процесу включення NFR в Agile призвело до низької якості цього емпіричного дослідження.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Вплив факторів трудового середовища на здоров'я та працездатність розробника програм

Безпека працівника нерозривно пов'язана з оточуючим її виробничим середовищем. Останнє характеризується породжуваними діяльністю людини об'єктами, явищами, фізичними, хімічними, біологічними та соціальними факторами, які прямо чи опосередковано впливають на самопочуття та стан здоров'я працюючих [58].

Людини може бути у безпеці тільки в такому стані виробничого середовища, коли виключена дія на неї небезпечних та шкідливих чинників.

Існує класифікація небезпечних та шкідливих факторів, яка розроблена для виробничих умов. Згідно з цією класифікацією небезпечні та шкідливі фактори за природою дії підрозділяються на 4 групи: фізичні, хімічні, біологічні та психофізіологічні [59], [60].

Будь-які фізіологічні, фізичні, хімічні чи емоційні впливи, будь то температура повітря, зміна атмосферного тиску або хвилювання, радість, сум можуть бути приводом до виходу організму зі стану динамічної рівноваги. Автоматично, на основі єдності різних механізмів регуляції здійснюється саморегуляція фізіологічних функцій, що забезпечує підтримку життєдіяльності організму на постійному рівні. При малих рівнях впливу подразника людина просто сприймає інформацію, що надходить ззовні. Вона бачить навколишній світ, чує його звуки, вдихає аромат різних запахів, сприймає дотиком і використовує у своїх цілях вплив багатьох факторів. При високих рівнях впливу виявляються небажані біологічні ефекти. Компенсація змін факторів довкілля виявляється можливою завдяки активації систем, відповідальних за адаптацію (пристосування).

Напруженість праці відображає навантаження на центральну нервову систему, психічні функції, характеризується обсягом сприйманої інформації, щільністю

сигналів, що надходять, станом аналізаторних систем, рівнем емоційної напруги і визначається ступенем напруги уваги. За цим показником працю поділяють на 4 групи (табл. 4.1)

Таблиця 4.1 – Класифікація робіт за напруженістю праці

Ступінь напруженості праці	Концентрація уваги
ненапружений	25 % часу роботи
мало напружений	50 % часу роботи
напружений	75 % часу роботи
дуже напружений	Більше 75 % часу роботи

За фізіологією, праця підрозділяється на:

– динамічну м'язову роботу, при якій м'язи різних м'язових груп поперемінно розтягуються і скорочуються (наприклад, при обертанні кривошипних рукояток);

– статичну м'язову роботу, при якій м'язи не рухаються.

При статичній роботі м'язи недостатньо поповнюються живильними речовинами, які переносяться кров'ю, і не звільняються від продуктів розпаду, що виникають при обміні речовин в організмі людини; це викликає хворобливе відчуття в м'язах і фізичну утому. Напруга при статичній роботі в 5 разів перевищує напругу, викликувану динамічною. При статичній роботі потрібно в 3 – 4 рази більше часу на відновлення енергії. Статична робота менш ефективна. При роботі в положенні стоячи ряд м'язів перебуває в постійній напрузі. При статичній роботі з навантаженням великої групи м'язів необхідно регулярно вводити перерви на відпочинок.

Основні принципи використання статичної роботи:

– статичне навантаження, що виникає при маніпулюванні органами керування, не повинні перевищувати 15 % максимального зусилля відповідної кінцівки при даній робочій позі оператора;

– при зусиллі перевищуючому 25% максимального зусилля, фізична утома спостерігається через 5 хв., а при зусиллі перевищуючому 50% максимального зусилля, м'язи витримують статичну напругу не більш 1 хв.

– робоче місце і робочі рухи повинні вибиратися таким чином, щоб обмежити статичну роботу до можливого мінімуму.

Для цього необхідно:

– обмежити до мінімуму виконання роботи в незручному положенні тіла або кінцівок;

– виключити виконання робіт у перебігу тривалого періоду часу в положенні руки розведені в сторони, підняті нагору, витягнуті вперед;

– обмежити тривалість утримання інструменту, матеріалу або перенесення вантажу;

– обмежити випадки збереження нерухомого положення тіла при виконанні робіт або дуже повільних робочих рухів руками.

Монотонна праця – це праця одноманітна, потребуюча від людини тривалого виконання однотипних простих операцій (монотонність дії) або безперервної концентрації уваги в умовах надходження малого обсягу професійно значимої інформації (монотонність обстановки).

При монотонній праці а організмі людини може розвитися комплекс фізіологічних і психологічних змін, відомий як стан монотонії. При виникненні стану монотонії – знижується продуктивність праці:

– збільшується брак продукції;

– зростає можливість прийняття невірних рішень; - одержання виробничих травм.

У результаті зменшується надійність людини, притупляється його пильність з можливими тяжкими наслідками в таких професіях як водії транспортних засобів, оператори пультів керування в енергетичній і хімічній промисловості, диспетчери аеропортів.

Серед факторів, що перешкоджають розвиткові монотонії, одне з ведучих місць займає ступінь функціональної робочої напруги, що включає.

– величину м'язових зусиль,

– темп роботи,

– ступінь її точності,

– наявність примусового темпу, ступенем складності і відповідальності, - рівень нервово-емоційної напруги.

Чим більше фізична чи тяжкість нервова напруженість праці, тим у меншому ступені монотонна, одноманітна праця приводить до розвитку стану монотонії.

До факторів, що сприяють розвиткові стану монотонії, відносяться:

– гіпокінезія, низька відповідальність

– фактори навколишнього оточення: постійний фоновий шум і вібрація, недостатнє освітлення, некомфортний мікроклімат, замкнутість робочого простору й одноманітність оформлення інтер'єру виробничих приміщень.

Стосовно монотонної діяльності люди поділяються на дві групи: монотофілів і монотофобів.

Для монотофілів характерні слабкий тип нервової діяльності, інертні нервові процеси, низькі показники по шкалі інтроверсії - екстраверсії, замкнутість, низький рівень нейротизму, низька тривожність. Монотофіли стійкі до розвитку монотонії, можуть виконувати монотонну роботу протягом тривалого часу .

Монотофоби володіють сильними процесами збудження, високою рухливістю нервових процесів, вираженої екстраверсію, високий рівень нейротизму, емоційну нестійкість, високу тривожність. Монотофоби схильні до розвитку монотонії при виконанні монотонної роботи.

4.2 Шкідливий вплив іонізуючого випромінювання

Іонізуючі випромінювання знаходять широке використання в різних галузях промисловості. Їх використовують для автоматичного контролю технологічних процесів, контролю якості виробів, зварних швів, структури металів тощо [61].

Для виробництва електроенергії на атомних електростанціях необхідне ядерне пальне, виробництво якого, починаючи від добування уранової руди і закінчуючи виготовленням та транспортуванням паливних елементів, призводить до опромінення персоналу. Незначні додаткові дози опромінення працівники отримують від таких техногенних джерел, як теплові електростанції (підвищена активність їх відходів та аерозолів), підприємств, які пов'язані з видобуванням та переробкою корисних копалин, а також різноманітних приладів та обладнання з джерелами випромінювання, що знаходять широке використання у промисловості і сільськогосподарському виробництві.

Основним документом, що встановлює радіаційно-гігієнічні регламенти для забезпечення прийнятих рівнів опромінення, є Норми радіаційної безпеки України (НРБУ-97).

НРБУ-97 регламентують опромінення людини джерелами іонізуючого випромінювання в умовах:

– нормальної експлуатації індустріальних джерел іонізуючого випромінювання;

– медичної практики;

– радіаційних аварій;

– опромінення техногенно-підсиленими джерелами природного походження.

Відповідно до цього НРБУ-97 встановлено чотири групи радіаційно-гігієнічних регламентів:

– перша – обмежує опромінення від ядерно-радіаційних об'єктів;

– друга – обмежує опромінення людей від медичних джерел;

– третя – обмежує опромінення в умовах радіаційних аварій;

– четверта – обмежує опромінення від техногенно підсилених джерел природного походження.

Враховуючи різнобічні наслідки опромінення людей іонізуючим випромінюванням, їх нормування здійснюється залежно від категорії людей, що

опромінюються, а також від чутливості органів тіла людини, на які діє іонізуюче випромінювання.

Виділяють наступні категорії:

А – особи з числа персоналу, які постійно чи тимчасово працюють безпосередньо з джерелами іонізуючого випромінювання;

Б – особи з числа персоналу, які безпосередньо не зайняті роботою з джерелами іонізуючого випромінювання, але у зв'язку з розташування робочих місць в приміщеннях та на промислових майданчиках об'єктів з радіаційноядерними технологіями можуть отримувати додаткове опромінення; В – все населення.

Частину населення, яке за своїми статевовіковими, соціально-професійними умовами, місцем проживання та іншими ознаками може отримувати найбільші рівні опромінення від даного джерела, прийнято виділяти як критичну групу.

Для осіб категорій А і Б НРБУ-97 встановлюються ліміти річних ефективних доз зовнішнього опромінення, а також ліміти річних еквівалентних доз зовнішнього опромінення окремих органів і тканин людини. Аналогічні ліміти вводяться і для критичних груп осіб категорії В. Ліміти дози опромінення наведені в табл. 4.2.

Таблиця 4.2 – Ліміти дози опромінення (мЗв/рік)

Назва лімітів	Категорія осіб, які зазнають опромінення		
	А	Б	В
$ЛД_E$ (ліміт ефективної дози)	20*	2	1
Ліміт еквівалентної дози зовнішнього опромінення:			
$ЛД_{lens}$ (для кришталика ока)	150	15	15
$ЛД_{skin}$ (для шкіри)	500	50	50
$ЛД_{e[trim}$ (для кистей та стіп)	500	50	-

Є також обмеження стосовно швидкості накопичення дози для жінок дітородного віку та вагітних жінок, підвищеного опромінення в непередбачуваних ситуаціях та інші.

Крім лімітів дози опромінення, встановлюють допустимі рівні (ДР): потужності дози зовнішнього опромінення, забруднення поверхонь, надходження радіонукліду через органи дихання тощо, які визначають виходячи із наведених лімітів дози опромінення.

З метою зниження рівнів опромінення населення Міністерство охорони здоров'я України запроваджує рекомендовані рівні медичного опромінення. При проведенні профілактичного обстеження населення річна ефективна доза не повинна перевищувати 1 мЗв.

Медичне опромінення – це опромінення працівників при медичних обстеженнях чи лікуванні. Опромінення повинно бути обґрунтованим і призначеним тільки лікарем для досягнення корисних діагностичних та терапевтичних ефектів, які неможливо отримати іншими методами діагностики та лікування.

Рекомендовані рівні медичного опромінення та детальні вимоги до обмеження і контролю за опроміненням пацієнтів регламентуються окремими спеціальними документами Міністерства охорони здоров'я України. При проведенні профілактичного медичного обстеження працівників річна ефективна доза не повинна перевищувати 1 мЗв.

Для радіометричного і дозиметричного контролю використовуються: дозиметри – для вимірювання зовнішніх потоків радіоактивного випромінювання; радіометри – для вимірювання рівнів забруднення навколишнього середовища; індивідуальні дозиметри – для індивідуального контролю.

Серед індивідуальних дозиметрів найбільше розповсюджені прилади, в яких використовують іонізаційні (за величиною іонізації середовища, через яке пройшло випромінювання) та фотографічні (за величиною опромінення фотографічної плівки іонізуючим випромінюванням) методи виміру.

У приладах для контролю потужності дози випромінювання широко застосовують іонізаційний та сцинтиляційний методи (за інтенсивністю світлових

спалахів, що виникають внаслідок люмінесценції в деяких речовинах під час проходження через них іонізуючих випромінювань).

При роботі з джерелами іонізуючих випромінювань здійснюють контроль і оцінку параметрів радіаційного фактора відповідно до НРБУ-97. При дотриманні контрольних рівнів умови праці на даному робочому місці оцінюються як допустимі. У разі їх перевищення оцінка шкідливості та небезпечності за радіаційним фактором здійснюється органами Держсанепіднагляду.

Засоби та заходи захисту від іонізуючих випромінювань поділяють на організаційні, технічні, санітарно-гігієнічні та лікувально-профілактичні.

Як правило, ефективний захист від іонізуючого випромінювання досягається при одночасному комплексному використанні зазначених заходів та засобів. При їх виборі враховуються особливості джерел випромінювання. Так, основними заходами, направленими на захист від альфа- та бета-випромінювань, є заходи, що націлені на недопущення накопичення альфа- і бета-активних ізотопів в організмі людини та забруднення шкіри: використання спеціального одягу та взуття, протипилових респіраторів, обезпилення повітря, вологе прибирання помешкань, недопущення вживання радіоактивно забруднених харчових продуктів, води та інші. При роботі з джерелами гама- та рентгенівського випромінювання захист персоналу досягається шляхом зниження активності джерел випромінювання, обмеження часу роботи з ними, збільшення відстані до джерел, екранування джерела іонізуючого випромінювання або зони знаходження людини.

ВИСНОВКИ

У цьому дослідженні порівнювався вплив методів розробки програмного забезпечення на якість з використанням проектів розробки програмного забезпечення, над якими працювала компанія, що складається з кількох підгруп, які надають фінансовим установам інноваційні рішення щодо послуг з іпотеки та кредитування.

У 2012 році компанія перейшла з Waterfall на метод Agile. Спочатку було обрано тридцять проектів цієї організації. Щоб мінімізувати упередження відбору, середні проекти (тобто 7-68 історій користувачів або робочі елементи), над якими працювали середні команди (тобто 3-11 членів команди) і тривалістю від 3 до 9 місяців, були визначені як проект для включення. Було відзначено дві важливі знахідки. По-перше, Agile-проекти мали значно більшу кількість дефектів. По-друге, коли дефекти були класифіковані на нефункціональні та функціональні вимоги, порівняння між двома групами показало, що Agile-проекти мають більше дефектів, пов’язаних з NFR, ніж у Waterfall.

Метою цієї кваліфікаційної роботи є не відстоювання того чи іншого методу. Основна мета — використовувати дані, отримані з кількох реальних проектів, щоб допомогти зрозуміти, чи негативно вплине застосування методу Agile на якість чи ні. Важливо зазначити, що оскільки міркування щодо бюджету та часу важко узагальнити, акцент було зосереджено на кількості дефектів як на більш стабільному параметрі для вимірювання.

Ці результати свідчать про те, що вибір Agile замість Waterfall може призвести до більшої кількості дефектів і, отже, до зниження якості, можливо, через недостатнє розуміння вимог. Це дослідження вносить свій внесок у процес прийняття рішень іншими компаніями, коли вони вибирають, який метод адаптувати та застосувати для них. Хоча деякі вважають, що процес Agile може бути універсальним засобом проти невдачі проекту розробки програмного забезпечення [46], це дослідження підтверджує це; спритність без ефективних вимог не може принести успіху [35].

В якості майбутньої роботи будуть проведені дослідження з використанням більшої вибірки, які можуть дати висновки, які можуть допомогти підтвердити поточні висновки. Іншою можливою майбутньою роботою є збір даних від різних організацій на основі розміру компанії, розміру команди та різного типу галузі.

СПИСОК ВИКОРСИТАНИХ ДЖЕРЕЛ

1. Agarwal, A.; Garg, N.K.; Jain, A., "Quality assurance for Product development using Agile," 2014 International Conference on Optimization, Reliability, and Information Technology (ICROIT), pp.44,47, 6-8 Feb. 2014 doi: 10.1109/ICROIT.2014.6798281"

2. Agile Alliance, Manifesto for Agile Software Development, 2001. Http://www.agilemanifesto.org/

3. Alsultanny, Y.A.; Wohaishi, A.M., "Requirements for Software Quality Assurance Model," Second International Conference on Environmental and Computer Science, 2009. ICECS '09., pp.19-23, 28-30 Dec. 2009 doi: 10.1109/ICECS.2009.43

4. Araujo, J.; Ribeiro, J.C., "Towards an aspect-oriented agile requirements approach", Eighth International Workshop on Principles of Software Evolution, 5-6 Sept. 2005, pp.140-143 doi: 10.1109/IWPSE.2005.31

5. B. Boehm, “Requirements That Handle Ikiwisi, COTS,and Rapid Change,” Computer, July 2000, pp. 99–102.

6. B. W. Boehm, "Verifying and Validating Software Requirements and Design Specifications," IEEE Software, vol. 1, pp. 75-88, 1984.

7. B. W. Boehm; J. R. Brown; M. Lipow, "Quantitative evaluation of software quality, " ICSE '76 Proceedings of the 2nd international conference on Software engineering, pp. 592-605.

8. Batool, A., Motla, Y.H., Hamid, B., Asghar, S., Riaz, M., Mukhtar, M., Ahmed, M.: Comparative study of traditional requirement engineering and Agile requirement engineering (2013) .

9. Begel, A.; Nagappan, N., "Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study," First International Symposium on Empirical Software Engineering and Measurement, 2007. ESEM 2007., pp.255-264, 20-21 Sept. 2007.

10. Benediktsson, O.; Dalcher, D.; Thorbergsson, H., "Comparison of software development life cycles: a multiproject experiment," Software, IEE Proceedings - , vol.153, no.3, pp.87,101, June 2006.

11. Bo Wei; Zhi Jin; Lin Liu, "A Formalism for Extending the NFR Framework to Support the Composition of the Goal Trees," Software Engineering Conference (APSEC), 2010 17th Asia Pacific , vol., no., pp.23,32, Nov. 30 2010-Dec.

12. Chapin, N., "Agile methods' contributions in software evolution," 20th IEEE International Conference on Software Maintenance, 2004. Proceedings. pp.522,, 11-14 Sept.

13. "Cysneiros, L.M.; Sampaio do Prado Leite, J.C., ""Nonfunctional requirements: from elicitation to conceptual models," IEEE Transactions on Software Engineering, vol.30, no.5, pp.328,350, May 2004

14. Farid, W.M.; Mitropoulos, F.J., "NORMATIC: A visual tool for modeling Non-Functional Requirements in agile processes," Southeastcon, 2012 Proceedings of IEEE , pp.1,8, 15-18 March 2012.

15. Farid, W.M.; Mitropoulos, F.J., "Novel lightweight engineering artifacts for modeling non-functional requirements in agile processes," Southeastcon, 2012 Proceedings of IEEE , pp.1,7, 15-18 March 2012.

16. Feng Ji; Sedano, T., ""Comparing extreme programming and Waterfall project results," 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), pp.482,486, 22-24, 2011.

17. Gallardo-Valencia, R.E.; Olivera, V.; Sim, S.E., "Are Use Cases Beneficial for Developers Using Agile Requirements?," Fifth International Workshop on Comparative Evaluation in Requirements Engineering, 2007. CERE '07. pp.11-22, 16-16 Oct. 2007.

18. Gallardo-Valencia, R.E.; Sim, S.E., "Continuous and Collaborative Validation: A Field Study of Requirements Knowledge in Agile", Second International Workshop on Managing Requirements Knowledge (MARK), pp.65-74, 1-1 Sept. 2009.

19. Gu Hongying; Yang Cheng, "A customizable agile software Quality Assurance model," 2011 5th International Conference on New Trends in Information Science and Service Science (NISS), vol.2, no., pp.382,387, 24-26 Oct. 2011.

20. Hamed, A.M.M.; Abushama, H., "Popular agile approaches in software development: Review and analysis," 2013 International Conference on Computing, Electrical and Electronics Engineering (ICCEEE), pp.160, 166, 26-28 Aug. 2013.

21. Hashmi, S.I.; Jongmoon Baik, "Software Quality Assurance in XP and Spiral - A Comparative Study," ICCSA 2007. International Conference on Computational Science and its Applications, 2007, vol., no., pp.367, 374, 26-29 Aug. 2007.

22. Hellmann, T.D.; Chokshi, A.; Abad, Z.S.H.; Pratte, S.; Maurer, F., "Agile Testing: A Systematic Mapping across Three Conferences: Understanding Agile Testing in the XP/Agile Universe, Agile, and XP Conferences," Agile Conference (AGILE), 2013, pp.32,41, 5-9 Aug. 2013.

23. Host, M.; Runeson, P., "Checklists for Software Engineering Case Study Research," First International Symposium on Empirical Software Engineering and Measurement, 2007. ESEM 2007., pp.479-481, 20-21 Sept. 2007.

24. Kassab, Mohamad, "An Empirical Study on the Requirements Engineering Practices for Agile Software Development," 2014, 40th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA) , pp.254,261, 27-29 Aug. 2014.

25. Lan Cao; Ramesh, B., "Agile Requirements Engineering Practices: An Empirical Study," Software, IEEE, vol.25, no.1, pp.60-67, Jan.-Feb. 2008.

26. Layman, L.; Williams, L.; Cunningham, L., "Exploring extreme programming in context: an industrial case study," Agile Development Conference, 2004, pp.32,41, 22-26 June 2004.

27. Lin Liu; Tong Li; Fei Peng, "Why Requirements Engineering Fails: A Survey Report from China," 2010 18th IEEE International Requirements Engineering Conference (RE), pp.317, 322, Sept. 27 2010-Oct. 1 2010.

28. Lopez, C.; Cysneiros, L.M.; Astudillo, H., "NDR Ontology: Sharing and Reusing NFR and Design Rationale Knowledge," Managing Requirements Knowledge, 2008. MARK '08. First International Workshop on , vol., no., pp.1,10, 8-8 Sept. 2008.

29. Macias, F “Empirical Assessment of Extreme Programming,” PhD thesis, Department of Computer Science, University of Sheffield, 2004.

30. Manjunath, K.N.; Jagadeesh, J.; Yogeesh, M., "Achieving quality product in a long term software product development in healthcare application using Lean and Agile principles: Software engineering and software development," 2013 International Multi-

Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), pp.26,34, 22-23 March 2013.

31. Martakis, A.; Daneva, M., "Handling requirements dependencies in agile projects: A focus group with agile software development practitioners," 2013 IEEE Seventh International Conference on Research Challenges in Information Science (RCIS), pp.1,11, 29-31 May 2013.

32. Ming Huo; Verner, J.; Liming Zhu; Babar, M.A., "Software quality and agile methods," Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004., pp.520-525 vol.1, 28-30 Sept. 2004.

33. Mitchell, S.M.; Seaman, C.B., "A comparison of software cost, duration, and quality for waterfall vs. iterative and incremental development: A systematic review," 3rd International Symposium on Empirical Software Engineering and Measurement, 2009. ESEM 2009., vol., no., pp.511,515, 15-16 Oct. 2009.

34. Mnkandla, E.; Dwolatzky, B., "Defining Agile Software Quality Assurance," International Conference on Software Engineering Advances, pp.36,36, Oct. 2006.

35. Orr, K., "Agile requirements: opportunity or oxymoron?," Software, IEEE , vol.21, no.3, pp.71-73, May-June 2004.

36. P. Clarke and R.V. O'Connor, "The Situational Factors that Affect the Software Development Process: Towards a Comprehensive Reference Framework," Information and Software Technology, vol. 54, no. 5, 2012, pp. 433-447.

37. Paetsch, F.; Eberlein, A.; Maurer, F., "Requirements engineering and agile software development," Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003. Proceedings, pp.308-313, 9-11 June 2003.

38. R. Grau, K. Lauenroth, B. Bereza, E. van Veenendaal, and S. van der Zee, "Requirements engineering and agile development-collaborative, just enough, just in time, sustainable," 2014.

39. Racheva, Z.; Daneva, M.; Buglione, L., "Supporting the Dynamic Reprioritization of Requirements in Agile Development of Software Products," Second International

Workshop on Software Product Management, 2008. IWSPM '08., pp.49,58, 9-9 Sept. 2008 doi: 10.1109/IWSPM.2008.7

40. Royce, Winston, W. "Managing the development of large software systems". Retrieved from http://www.cs.umd.edu/ class/spring2003/ cmsc838p/Process/ waterfall.pdf

41. Savolainen, J.; Kuusela, J.; Vilavaara, A., "Transition to Agile Development - Rediscovery of Important Requirements Engineering Practices," Requirements Engineering Conference (RE), 2010 18th IEEE International, pp.289-294, Sept. 27 2010-Oct. 1 2010.

42. Scharff, C., "Guiding global software development projects using Scrum and Agile with quality assurance," 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), pp.274,283, 22-24 May 2011.

43. Singh, B.; Kannojia, S.P., "A Review on Software Quality Models," 2013 International Conference on Communication Systems and Network Technologies (CSNT), pp.801, 806 , 6-8 April 2013 doi: 10.1109/CSNT.2013.171

44. Sommerville,I; Software Engineering (9th Edition) Harlow, England; New York Addison- Wesley, Mar 3 2010

45. Soundararajan, S.; Arthur, J.D., "A Soft-Structured Agile Framework for Larger Scale Systems Development," 16th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, 2009. ECBS 2009., pp.187-195, 14-16 April 2009.

46. STANDISH GROUP - 2013 The CHAOS Manifesto–Think Big, Act Small

47. Sureshchandra, K.; Shrinivasavadhani, J., "Moving from Waterfall to Agile," Agile, 2008. AGILE '08. Conference , vol., no., pp.97,101, 4-8 Aug. 2008.

48. Taehoon Um; Neunghoe Kim; Donghyun Lee; Hoh Peter In, "A Quality Attributes Evaluation Method for an Agile Approach," Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on , pp.460,461, 23-25 May 2011.

49. Thayer, R.H., and M. Dorfman: Software Requirements Engineering. 2d ed., IEEE Computer Society Press (1997)

50. Tsun Chow, Dac-Buu Cao, “A survey study of critical success factors in agile software projects,” Journal of Systems and Software, v.81 n.6, p.961-971, June, 2008.

51. Vijayasarathy, L.; Butler, C., ""Choice of Software Development Methodologies - Do Project, Team and Organizational Characteristics Matter?,"" Software, IEEE , vol.PP, no.99

52. Waldmann, B., "There's never enough time: Doing requirements under resource constraints, and what requirements engineering can learn from agile development," Requirements Engineering Conference (RE), 2011 19th IEEE International, pp.301,305, Aug. 29 2011-Sept. 2, 2011.

53. Wieringa, R., "Towards a unified checklist for empirical research in software engineering: first proposal," 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012), pp.161,165, 14-15 May 2012.

54. Yi Liu; Zhiyi Ma; Rui Qiu; Hongjie Chen; Weizhong Shao, "An Approach to Integrating Non-functional Requirements into UML Design Models Based on NFR-Specific Patterns," 2012 12th International Conference on Quality Software (QSIC), pp.132,135, 27-29 Aug. 2012.

55. Yin, R. K. “Case study research: Design and Methods (3rd Edition)”. Thousand Oaks, CA: Sage. 2003.

56. Харченко, Олександр Григорович, Василь Володимирович Яцишин, and Ігор Едуардович Райчев. "Інструментальний засіб розробки та комунікації вимог якості до програмних систем." (2010).

57. Харченко, О., and В. Яцишин. "Розробка та керування вимогами до програмного забезпечення на основі моделі якості." Вісник Тернопільського національного технічного університету 66,№ 1 (2009): 201-207.

58. Державні санітарні норми та правила "Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу" // Офіційний вісник України – 2014. – № 41.– С. 95-132.

59. Крушельницька Я. В. К 84 Фізіологія і психологія праці: Підручник. – К.: КНЕУ, 2003. – 367 с.

60. Батлук В.А., Гогіташвілі Г.Г. та ін. Охорона праці в галузі телекомунікацій. – Львів: Афіша, 2003. – 320с.

61. Методичні рекомендації для проведення атестації робочих місць за умовами праці. Затверджено міністром праці України 1.09.1992 р, постанова № 41.

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

IX НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

8–9 грудня 2021 року

ТЕРНОПІЛЬ
2021

УДК 004.42
Р. Войтюк, Ю. Тарасовський
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ЗАДАЧА ВИБОРУ ПРОГРАМНОЇ АРХІТЕКТУРИ ПРИ ЗМІНІ ВИМОГ

UDC 004.42
R. Voitiuk, Yu. Tarasovskyi

SOFTWARE ARCHITECTURE CHOOSING FOR CHANGING REQUIREMENTS

Сучасні гнучкі технології програмування по суті є ітераційними. При виконанні поточної ітерації можуть вноситися зміни у вимоги, або обмеження, що потребуватиме внесення відповідних змін у розділи проєкту, в тому числі і в розділ архітектури. Також в процесі експлуатації програмної системи (ПС) відбуваються зміни вимог предметної області, що викликає необхідність внесення змін в ПС, а тому в першу чергу необхідні зміни програмної архітектури (ПА), оскільки вона визначальним чином впливає на якість ПС. Вибір архітектури здійснюється з множини альтернатив, які конструюються на основі функціональних вимог із стандартних компонентів. Для вибору найкращого варіанта архітектури обчислюються їх оцінки по окремих критеріях якості, а потім на основі отриманих оцінок виконується багатокритерійний вибір архітектури. Задача оцінювання альтернатив по окремим критеріям якості найбільш ефективно розв'язується використанням методу аналізу ієрархій Сааті (MAI) або його модифікованого варіанта (ММАІ). Суттєвим недоліком застосування MAI є обмежена кількість альтернатив, які можна оцінювати одночасно ($n \leq 7 \pm 2$).

В роботах [1], [2] розглянуті питання застосування модифікованого MAI до задачі оцінювання альтернативних варіантів архітектури програмних систем при великій кількості альтернатив. В цих методах відносна оцінка альтернатив визначається з використанням експертної інформації і при зміні вимог до ПС на черговій ітерації проєктування потрібно повторно проводити експертне оцінювання та розрахунки оцінок критеріїв альтернатив. Оскільки в ітераційних технологіях проєктування ПС процеси можуть виконуватись одночасно на декількох стадіях життєвого циклу з використанням базового варіанта архітектури, то його зміна потребуватиме внесення коректив в декілька розділів проєкту.

Для зменшення об'єму необхідних змін в проєкті, пов'язаних із зміною вимог до ПС, пропонується використати процедуру корекції критеріїв якості базової архітектури та оптимізації цієї корекції. Оптимізація заміщення є задачею багатокритерійної оптимізації. В якості критерія, який оптимізується, пропонується використати нелінійну скалярну згортку. В ній оптимізується цільова функція, яка залежить від міри «напруженості ситуації», котра визначається близькістю значень критеріїв до своїх обмежень. Для формалізації процесу визначення ваг критеріїв використовується ітераційна процедура симплекс-планування. Отримані оптимальні значення корекції критеріїв використовуються для модифікації архітектури ПС.

Література

1. Харченко О.Г. Метод багатокритеріальної оптимізації програмної архітектури на основі аналізу компромісів [Текст] / Харченко О.Г., Боднарчук І.О., Галай І.О. // Інженерія програмного забезпечення. – К.: НАУ.-2012. – № 3–4 (11–12). – с. 5 – 11.
2. Kharchenko A. The method for comparative evaluation of software architecture with accounting of trade-offs/ Alexander Kharchenko, Ihor Bodnarchuk, Vasyl Yatcyshyn // American Journal of Information Systems. V. 2, No. 1. 2014. – P. 20-25. Available online at http://pubs.sciepub.com/ajis/2/1/5