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: Проблеми інженерії вимог для гнучких технологій розробки програмного забезпечення

Виконав(ла): студент(ка) 6 курсу, групи СНм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

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

Керівник Никитюк В.В.
(підпис) (прізвище та ініціали)

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

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

Рецензент Петрик М.Р.
(підпис) (прізвище та ініціали)

Тернопіль
2022

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 Шевчук Юрій Володимирович
(прізвище, ім'я, по батькові)

1. Тема роботи **Проблеми інженерії вимог для гнучких технологій розробки програмного забезпечення**

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

Затверджені наказом ректора від «*22*» *листопада* 2022 року № *4/7-948*

2. Термін подання студентом завершеної роботи 19 грудня 2022 р.

3. Вихідні дані до роботи Літературні джерела з тематики роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
ВСТУП; 1 ЗАДАЧІ ІНЖЕНЕРІЇ ВИМОГ В AGILE-ПРОЕКТАХ; 1.1 Гнучка розробка програмного забезпечення та розробка вимог; 1.2 Проблеми масштабування Agile; 1.3 Практики розробки вимог у гнучкій розробці програмного забезпечення; 2 ОСОБЛИВОСТІ РЕАЛВЗАЦІЇ ПРОЦЕСУ ІНЖЕНЕРІЇ ВИМОГ ДЛЯ КОМПАНІЙ РІЗНОГО РОЗМІРУ; 2.1 Проблеми розробки вимог для гнучких проектів малого та середнього масштабу; 2.2 Інженерні виклики вимог для великомасштабних гнучких проектів; 2.3 Відомі виклики, їх причини та потенційні рішення при роботі з вимогами в гнучких проектах; 3 ЗАГАЛЬНІ РЕКОМЕНДАЦІЇ ЩОДО ВДОСКОНАЛЕННЯ ПРОЦЕСУ AGILE-РОЗРОБКИ ВИМОГ; 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ; 4.1 Методи та засоби психофізіологічного розвантаження як допоміжний процес в розробці ПЗ; 4.2 Попередження аварій на виробництвах із застосуванням хлору; ВИСНОВКИ; СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ; ДОДАТКИ

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	Мацюк О.В., к.т.н., доц.		
Безпека в надзвичайних ситуаціях	Клепчик В.М., ст. викл.		

7. Дата видачі завдання 21 вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	14.11.22-15.11.22	*Виконано*
2.	Підбір наукових джерел по темі роботи	16.11.22-20.11.22	*Виконано*
3.	Переклад та опрацювання наукових джерел	20.11.22-23.11.22	*Виконано*
	по темі кваліфікаційної роботи		
4.	Виконання дослідження щодо теми	24.11.22-10.12.22	*Виконано*
	Кваліфікаційної роботи		
5.	Оформлення першого розділу	11.12.22-12.10.22	*Виконано*
6.	Оформлення другого розділу	12.12.22-13.12.22	*Виконано*
7.	Оформлення третього розділу	13.12.22-14.12.22	*Виконано*
8.	Виконання завдання до підрозділу «Охорона праці»	08.12.22-09.11.22	*Виконано*
9.	Виконання завдання до підрозділу «Безпека в	10.12.22-12.12.22	*Виконано*
	надзвичайних ситуаціях»		
10.	Оформлення кваліфікаційної роботи	12.12.22-31.12.22	*Виконано*
11.	Нормоконтроль	14.12.22-15.12.22	*Виконано*
12.	Перевірка на плагіат	15.12.22	*Виконано*
13.	Попередній захист кваліфікаційної роботи	16.12.22	*Виконано*
14.	Захист кваліфікаційної роботи	20.12.2022	

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

Керівник роботи _____ Никитюк В.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

"Проблеми інженерії вимог для гнучких технологій розробки програмного забезпечення" // Кваліфікаційна робота освітнього рівня «Магістр» // Шевчук Юрій Володимирович // Тернопільський національний технічний університет ім. І. Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНм-61 // Тернопіль, 2022 // с. – 45, рис. – 5, табл. – 3, джерел – 22.

Ключові слова: Agile, інженерія вимог, гнучка розробка, управління проектами, масштабування Agile

Гнучка розробка програмного забезпечення досить поширена завдяки своїм перевагам. Це в першу чергу властиве для невеликих проектів. Розробка вимог (Requirements Engineering – RE) має вирішальне значення, оскільки в кожному проєкті з розробки програмного забезпечення «Вимоги» відіграють життєво важливу роль. Саме інженерія вимог до програмного продукту найчастіше є тим пунктом, що заважає впровадити гнучкі методології в проєкт повністю.

Робота представляє огляд гнучких моделей з точки зору розробки вимог. Висновки та результати можуть бути дуже корисними для індустрії програмного забезпечення для вдосконалення процесу розробки, а також для дослідників, які хочуть працювати далі в цьому напрямку.

ANNOTATION

“Problems of Requirements Engineering for Agile Software Development Techniques” // Master’s degree qualification paper // Shevchuk Yurii Volodymyrovytch // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Computer Science Department, group СНм-61 // Ternopil, 2022 // p. – 45, fig. – 5, tables – 3, references – 22.

Key words: Agile, requirements engineering, adaptive development, project management, Agile scaling

Agile software development is quite common due to its advantages. This is primarily typical for small projects. Requirements Engineering (RE) is critical, as Requirements play a vital role in every software development project. It is the engineering of the requirements for the software product that is most often the point that prevents the implementation of flexible methodologies in the project completely.

This paper presents an overview of flexible models from the point of view of requirements development. The findings and results can be very useful for the software industry to improve the development process, as well as for researchers who want to work further in this direction.

ЗМІСТ

ВСТУП

Актуальність задачі.

Agile – це адаптивний підхід до розробки, і високий рівень успіху проєктів зробив його дуже популярним у спільноті розробників програмного забезпечення. Ці технології висвітлені також в роботах науковців ТНТУ, зокрема в [1]. Адаптивна розробка програмного забезпечення (Adaptive Software Development – ASD) найкраще працює з традиційними підходами та практиками розробки через багато причин, як-от потреби клієнтів, що швидко змінюються, термінові зміни бізнес-процесів тощо. Вибір правильного SDLC (життєвого циклу розробки програмного забезпечення) під час розробки програмного рішення є дуже важливим завданням. Будь то водоспадна модель, V-модель або Agile, усі вони мають свої переваги та недоліки [2]. Вибір підходу SDLC в основному залежить від характеру продукту, що розробляється.

У розробці програмного забезпечення «Інженерія вимог» (RE) відіграє життєво важливу роль, оскільки весь успіх програмного забезпечення залежить від правильності вимог, зібраних на етапі розробки [3, 4]. Отже, коли природа вимог є динамічною, то завдання зібрати, проаналізувати, зрозуміти та керувати вимогами є непростим [5]. Тому моделі роботи з вимогами у гнучких проєктах є актуальною задачею.

Мета роботи.

Метою дослідження є виконання огляду літературних джерел та встановлення особливостей процесу розробки вимог до програмного продукту в проектах з гнучкою організацією.

Об’єкт дослідження: Процеси управління вимогами до програмного забезпечення зав гнучкими методологіями.

Предмет дослідження: моделі управління проєктами по гнучких методологіях.

Наукова новизна отриманих результатів. Запропоновано моделі роботи з вимогами для компаній різних розмірів в Agile методологіях

Практичне значення отриманих результатів. Ця робота представляє детальний огляд і представляє відповідні висновки та пропозиції. Дослідження спрямовані на з’ясування проблем та викликів, з якими стикаються під час проектування вимог маломасштабних, середньомасштабних та великих гнучких проектів розробити кілька можливих пропозицій щодо покращення загального процесу RE в Agile на основі аналізу та оцінки літератури, щоб запропонувати пропозиції щодо майбутньої роботи.

Апробація результатів та особистий внесок здобувача. Основні положення роботи доповідались, розглядались та обговорювались на науковій конференції Тернопільського національного технічного університету імені Івана Пулюя. Результати кваліфікаційної роботи опубліковані у тезах студентської наукової конференції "Інформаційні моделі системи та технології – 2022", яка проводились у ТНТУ.

1 ЗАДАЧІ ІНЖЕНЕРІЇ ВИМОГ В AGILE-ПРОЕКТАХ

Міжнародне дослідження CHAOS групи Standish Group виявило багато причин, проблем і факторів невдачі проектів з розробки програмного забезпечення. Дослідження показують, що проблеми розробки проекту становлять до 37% від якості вимог, як показано на рисунку 1.1.

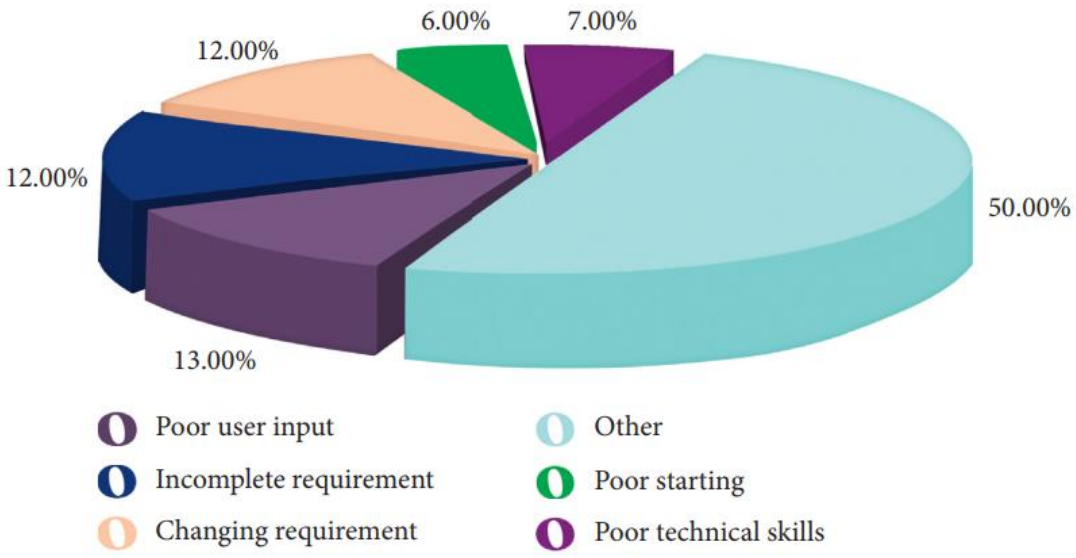

Рисунок 1.1 – Причини провалу проекту

Розглянуто проблеми та рішення відстеження вимог у ASD та деякі рекомендації для гнучкого проектування вимог (Adaptive Requirements Engineering – ARE). Більшість проектів програмного забезпечення зазнають невдачі через погане управління вимогами та через загальні проблеми, які спричиняють невдачу проекту.

Компанії з розробки програмного забезпечення використовують гнучкі підходи, коли невелика команда дуже тісно співпрацює з клієнтами для створення високоякісного програмного забезпечення з частими ітераціями та зворотним зв’язком. Якщо характер проекту невеликий, гнучка методологія працює дуже добре та створює ефективні продукти. Щоб отримати переваги гнучкості у випадку складних програмних продуктів і великих компаній, основні

припущення Agile напряму не працюватимуть, і існує потреба задовольнити додаткові обмеження.

Існує багато сценаріїв, коли компанії, які дотримуються повної або часткової практики гнучкості для великомасштабних або розподілених проектів, мали позитивний вплив використання гнучкості у великомасштабних проектах програмного забезпечення. Наприклад, Yahoo, Google, Microsoft, Siemens, CNBC, AOL та інші використовують Agile у своїх проектах розробки [7]. Існує багато принципів і вказівок щодо використання гнучкої методології для складної та великомасштабної розробки програмного забезпечення, наприклад [8].

Численні дослідження показали, що гнучкий підхід довів ефективність для масштабних проектів. В роботі [9] описано причини зміни вимог і класифіковано їх на дві категорії: випадкові та істотні. Існує цілий ряд інших робіт з дослідженнями роботи з вимогами в Agile проектах.

Незважаючи на те, що було проведено численні дослідження гнучкої розробки та впровадження гнучких практик, існує обмежена кількість досліджень проблем, з якими стикається RE, використовуючи гнучку стратегію розвитку та їх вирішення. Переважно ці огляди десь і якось носять вибірковий та упереджений характер.

Зазначені вище обмеження спонукали провести огляд літератури на тему «Проблеми розробки вимог у гнучкій розробці програмного забезпечення». Далі в роботі подається обговорення гнучкої розробки програмного забезпечення та розробки вимог. Обговорюються також проблеми розробки вимог для малих, середніх гнучких проектів та для великомасштабних гнучких проектів. Потім матеріал присвячений висвітленню проблемам, їх причинам і потенційним рішенням.

Відповідно до літератури, багато промислових проектних груп працюють у гнучкому середовищі, щоб мати швидку доставку високоякісного програмного забезпечення як для малих, так і для великих проектів. Схема запропонованої методики представлена на рисунку 1.2.

1.1 Гнучка розробка програмного забезпечення та розробка вимог

Сама по собі Agile – це не метод, а підхід, заснований на маніфесті, який в основному зосереджується на 12 принципах і 4 основних цінностях. Цінності гнучкого маніфесту – це робоче програмне забезпечення з повною документацією, люди та взаємодія між інструментами та процесом, співпраця з клієнтами та швидке реагування на зміни. Гнучкі методи розробки суттєво відрізняються від традиційних планових підходів тим, що зосереджені на продуктивності, а не на строгому дотриманні процесів. Відповідно до [10], понад 90% практиків гнучкої методології використовують історії користувачів (users stories), щоб фіксувати вимоги.

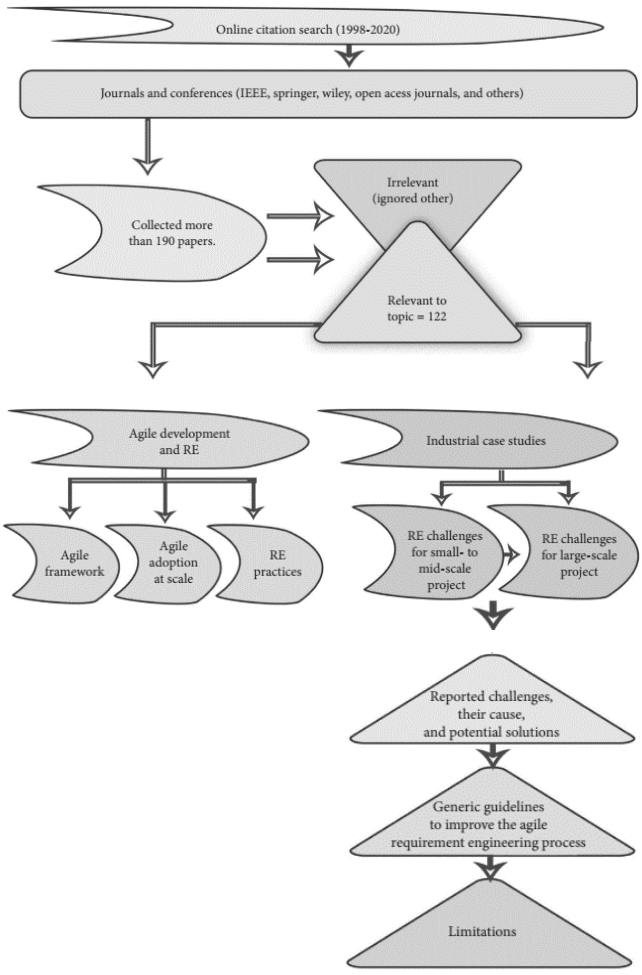

Рисунок 1.2 – Методика запровадження Agile для великих компаній

ASD – це адаптивний процес із меншою кількістю документації, обмеженими ресурсами та висококваліфікованими й мотивованими членами команди малого та середнього розміру. Застосування гнучких методів дозволяє розробляти надійне та корисне програмне забезпечення за допомогою кількох ітерацій. Agile розглядає питання традиційної моделі розвитку RE, оскільки основні виклики пов’язані з вимогами та зміною бізнес-процесу. Гнучка розробка долає проблеми послідовної моделі, але все ж є деякі недоліки, пов’язані з нею.

Нижче наведено огляд цих адаптивних моделей FDD (розробка, керована функціями), екстремальне програмування (XP) і SCRUM та адаптивного моделювання (AM) з точки зору розробки вимог.

AM – це техніка, яка надає деякі вказівки, пов’язані з моделями проектування, і використовується для мінімального збереження документації та моделей. Підхід до розробки вимог на основі мозкового штурму, підтримується деякими додатками AM.

FDD – це 5-етапна процедура з деякими конкретними критеріями входу та виходу, головним чином зосередженими на фазах будівництва та проектування, списку функцій побудови та плануванні, які ітеративно проектуються та створюються за кроками функцій, де функція представляє клієнтську ціннісну функцію. Продукт розробляється з точки зору окремих ознак замість цілого або повного продукту.

XP – це відомий гнучкий метод, який використовується для успішної розробки програмного забезпечення та підтримує легкість, комунікацію, зворотній зв’язок і простоту, незважаючи на постійні зміни вимог. Звичайні практики XP такі: парне програмування, швидкі ітерації з невеликими випусками, швидкий зворотній зв’язок, тісна взаємодія та участь клієнтів тощо. XP фокусується на кодуванні та розробці замість збору вимог. У XP команда передає відповідальність за зручність використання продукту зацікавленим сторонам.

SCRUM – це гнучкий, адаптивний підхід до управління проектами з 30-денним циклом спринту (як правило). З метою координації та інтеграції щоденно проводяться 15-хвилинні зустрічі. У SCRUM розроблений продукт завжди знаходиться в робочому стані і після кожної ітерації збільшується лише функціональність продукту.

Такі види діяльності, як виявлення, аналіз, конкретизація та документування, мають велике значення. Пріоритезація вимог у гнучкій розробці здійснюється за допомогою точок зору. Високоякісні проекти розробляються за менший час за допомогою гнучких методологій розробки, включаючи ітераційну розробку, збільшення проекту, адаптацію та співпрацю.

1.2 Проблеми масштабування Agile

Великий проект може бути визначений за деякими критеріями, які використовує компанія, або за допомогою абсолютних показників, таких як кількість залучених розробників, розмір бюджету, складність або кількість команд розробників. Позитивні результати та рівень успіху гнучкого впровадження в невеликих проектах викликали інтерес команд розробників програмного забезпечення та промисловості до впровадження Agile у великих проектах, оскільки гнучке впровадження швидко замінює традиційні методи.

Прийняття широкомасштабного впровадження проекту програмного забезпечення може спричинити проблеми з управлінням проектом. Незважаючи на ці проблеми, швидкість впровадження Agile у великому масштабі швидко зростає [11]. Було вивчено інформацію про 108 проектів програмного забезпечення, і їхні результати демонструють успішність впровадження гнучкої методології в малих, середніх і великих проектах [12] . Результати рівня гнучкого впровадження проектів показано на рисунку 1.3.

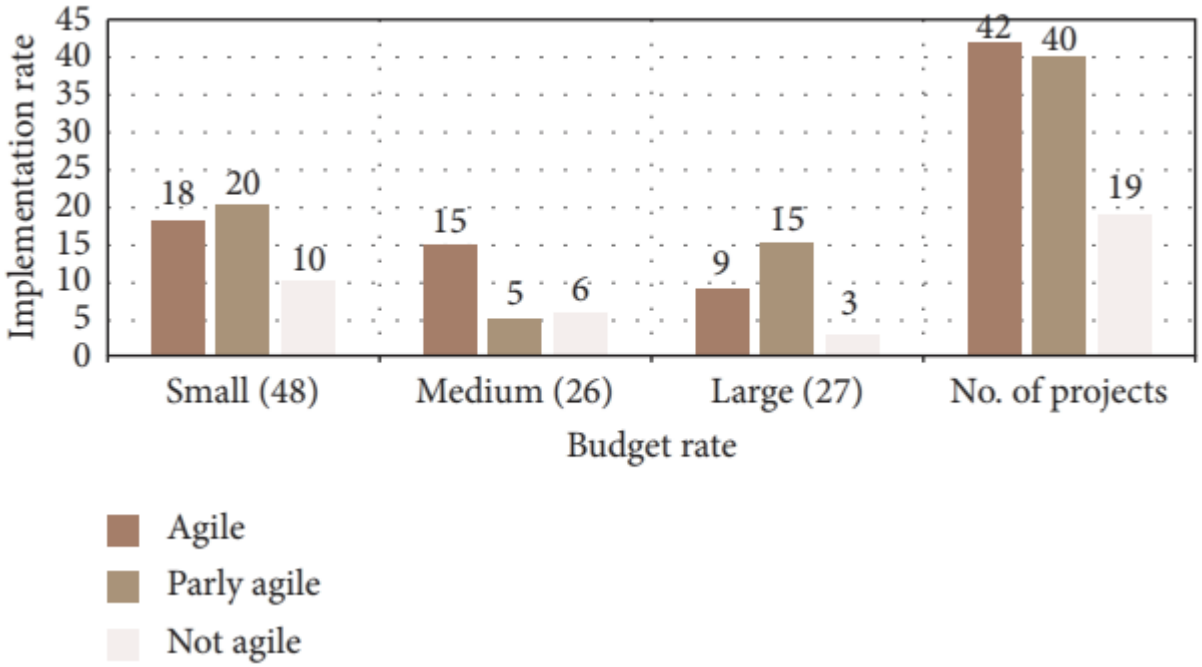

Рисунок 1.3 – Рівень впровадження Agile у порівнянні з проектами

З іншого боку, застосування Agile в великому масштабі має певні обмеження в деяких випадках, тобто SCRUM підходить для невеликих проектів з меншою кількістю членів команди, і коли він використовується в масштабний проект, це робить розробку менш ефективною. В роботі [13] визначили 21 мотивацію для впровадження Agile у великих масштабах. Крім того, їх аналіз і класифікацію, наприклад, на критичні фактори, було зроблено на основі визначених критеріїв. Виявлені мотиватори підтверджуються опитуваннями, анкетами та емпіричним дослідженням у присутності та за допомогою практикуючого спеціаліста з розробки програмного забезпечення та експертів. Узагальнені вище факти підтверджують і створюють шлях до широкомасштабного впровадження Agile в програмних проектах.

1.3 Практики розробки вимог у гнучкій розробці програмного забезпечення

Відповідно до [14], RE у гнучкій розробці здійснюється протягом усього циклу розробки, але в невеликих сесіях порівняно з традиційною поетапною інженерією вимог. Нижче наведено деякі поширені гнучкі практики RE:

– щоб отримати користь від гнучкого процесу, зацікавлені сторони та члени команди повинні спілкуватися безпосередньо;

– для успішного проекту та уникнення невдач головним фактором є наявність послідовних і пріоритетних вимог замовника;

– виявлення вимог, визначення пріоритетів і управління змінами, включаючи оновлення функцій, є важливими;

– необхідне безперервне планування, щоб впоратися з новими та мінливими вимогами, а діяльність з розробки вимог виконується на основі спільних концепцій;

– типовим є використання прототипу, щоб отримати відгук від клієнта для уточнення вимог і визначення пріоритетів.

Раннє тестування продукту, рефакторинг, парне програмування, невеликі випуски та залучений до розробки клієнт є фундаментальними практиками, які важливі для розуміння та впровадження Agile. Окрім цього, розбиття великих артефактів на дрібніші, написання тільки основних тестів, комунікація, об’єктно-орієнтоване проектування та використання передових інструментів для швидкого циклу розробки вважаються важливими навичками гнучкої розробки. Загальні виклики, з якими стикаються в гнучких проектах під час процесу розробки вимог разом із їх діяльністю, визначені та представлені в таблиці 1.1.

Таблиця 1.1 – Короткий перелік проблем, з якими стикаються під час гнучкої розробки вимог

Розробка вимог (етап)	**Діяльність в Agile**	**Можливі проблеми**	**Їх вплив**
Викликання	Інтерв'ю та анкетування	Обсяг проблеми, нерозуміння, двозначність мови створюють недоліки	Неправильно визначені вимоги впливають на аналіз вимог
Викликання	Мозковий штурм	Груповий мозковий штурм	Неоднозначний або погано визначений
Викликання	Прототипування	Безпека, масштабованість і надійність	Проблема з обслуговуванням
Аналіз	Розстановка пріоритетів	Конфліктні ідеї	Неоднозначність
Документація	User Stories	Нерелевантні історії користувачів, відсутність представників клієнтів	Введення в оману
Документація	Беклог продукту та індексні картки	Мінімум документації	Втрата знань
Перевірка	Відгуки клієнтів	Відсутність належних методів та інструментів перевірки, проблеми зі створенням прототипів	Низька якість
Управління	Зміни та контроь версій	Вибір відповідного інструменту	Втрата часу
Управління	Трасування вимог	Немає ефективного способу управління вимогами та відстеження	Відсутність трасування

2 ОСОБЛИВОСТІ РЕАЛВЗАЦІЇ ПРОЦЕСУ ІНЖЕНЕРІЇ ВИМОГ ДЛЯ КОМПАНІЙ РІЗНОГО РОЗМІРУ

2.1 Проблеми розробки вимог для гнучких проектів малого та середнього масштабу

Гнучка розробка долає проблеми послідовної моделі, але все ж є недоліки, пов’язані з нею. Проблеми інженерії вимог полягають у несумісному інтерфейсі, нехтуванні нефункціональними вимогами та відсутності однозначності. Гнучкі підходи не мають жодної належної структури, за допомогою якої вимоги користувачів належним чином перераховуються або документуються, і через зміну вимог необхідно виконувати великий обсяг роботи неодноразово.

Іноді через зміну потреб замовника розроблений продукт у попередній ітерації викликає проблеми сумісності інтерфейсу. Немає правила чи методики збору та оцінки нефункціональних вимог. Якість продукту перевіряється зворотним зв'язком через користувача після завершення ітерації. Загалом проблеми можна сформулювати у вигляді наступного списку.

1. У виявленні вимог: недостатня ясність, пріоритетність вимог і визначення обсягу проблеми.
2. В управлінні вимогами: питання визначення пріоритетів.
3. У вимогах до документації: відсутність належної документації та відсутність представника клієнта.
4. У перевірці вимог: немає методів та інструментів для процесу перевірки.
5. В управлінні вимогами: відсутність управління змінами вимог і відповідних інструментів управління.

Відсутність належної документації може створити деякі проблеми для команди; отже, варто призначити в команді деяких людей для створення невеликої кількості документації, використання комп’ютерних інструментів для моделювання та розробки методів зворотного проектування. Менша пряма

комунікація та недостатня документація спричиняють проблеми з відстеженням вимог і менше зворотного зв'язку та взаємодії з клієнтом, обробка змін вимог спричиняє більше роботи та обсяг часу виходить за плановий показник, а творчі та інноваційні вимоги в самій гнучкій системі викликають труднощі, відсутні і суперечливі вимоги, що спричиняє велику кількість ітерацій і відсутність уваги на нефункціональних вимогах.

Проблеми, з якими стикаються при розробці вимог при використанні SCRUM, полягають у відсутності участі команди в початковій оцінці проекту, вплив неоднозначних вимог на якість і графік, мінімум документації, різні рівні абстракції та неузгодженість історій користувачів у беклозі продукту після призначення власника продукту. Через зміну вимог гнучкий проект стає дорогим, тому вартість проекту та вартість обслуговування збільшуються. Мінімальна документація в процесі збору вимог створює такі проблеми, як відсутність допомоги персоналу та не забезпечує підтримки методів зворотного проектування. Відсутність документації створює інші проблеми, коли проект передається на обслуговування, коли змінюється персонал або коли наймається нова команда чи персонал.

В гнучкому підході RE більше зосереджується на перевірці вимог, але немає формальної моделі для перевірки детальних вимог в Agile. Для визначення пріоритетів вимог єдиним критерієм є бізнес-цінність, оскільки час виходу на ринок є важливим фактором, який може створити серйозні проблеми згодом. У Agile RE використання прототипування створює неправильне розуміння в свідомості зацікавлених сторін щодо швидкості розробки. Забагато впровадження на початковому етапі створення прототипів спричиняє проблеми з якістю, спричинені практикою повторного використання коду в прототипі, і зацікавлені сторони можуть не прийняти інші цикли розробки, які є більш масштабованими та надійнішими. Проблеми RE також полягають у чіткій комунікації та обміні інформацією, знаннях домену та меншій кількості документації щодо вимог до програмного забезпечення.

Основна причина полягає в тому, що клієнт і вся команда більше не доступні для щоденних особистих зустрічей. Регулярний зв’язок неможливий, тому мінімальні специфікації спричиняють багато проблем, як-от труднощі для команди із забезпечення якості та втрату знань щодо домену чи продукту.

В [15] викладено опис викликів та проблем SCRUM (модель гнучкої розробки) на прикладі компанії, яка використовувала SCRUM у своїх веб-проектах малого та середнього розміру. У повідомленнях про проблеми розробник сказав, що частина складного програмного коду потребує додаткових пояснень і внесення змін. Тоді як багато розробників казали, що це легко зрозуміти без документації. Розробники зазначили питання щодо важливості специфікації вимог, оскільки в команді немає спільних знань про продукт, і якщо особа, яка володіє всією інформацією, покине компанію та якийсь інший важливий член команди не зможе замінити цю людину.

Зацікавлені сторони не беруть участі в процесі прийняття рішень, а клієнт не завжди доступний для регулярних зустрічей, що призводить до розриву комунікації. Гнучкий підхід спирається на основне припущення, що отримання та виявлення всіх вимог здійснюється від користувача регулярно і постійно, оскільки вони розвиваються з часом, таким чином клієнт змінює свої вимоги та їх розуміння. Таким чином, ніхто не знає повних вимог до продукту на початку проекту, що спричиняє відсутність інтерфейсу між вимогами та викликає додаткові переробки в наступних ітераціях.

Іншими повідомленими проблемами є відсутність діяльності з RE та відсутність уваги до нефункціональних вимог. Слід визначити проблеми, пов’язані з нефункціональними вимогами, і команда повинна прийняти нефункціональні вимоги для покращення загальної якості продукту на кожній ітерації; інакше це також може спричинити проблеми з безпекою та масштабованістю, які сильно вплинуть на остаточну версію продукту.

У пріоритезації вимог користувачі відіграють важливу роль як історія білдів, яка включає важливі технічні деталі та створюється у співпраці користувачів. Таким чином, нездатність користувача, як-от користувачі, не

можуть точно сказати, що їм потрібно і що вони насправді хочуть сказати через високий тиск влади, факти навколишнього середовища або вагання та меншу представництво в їхній групі, створюють проблеми.

У Agile клієнти не беруть участі в процесі прийняття рішень, і це створює проблеми, пов’язані з пріоритезацією вимог. Пріоритезація вимог у гнучкому підході після кожної ітерації є складною за відсутності будь-якої техніки, і коли кожна вимога проекту вимагає конкретного розгляду кожного аспекту, це викликає проблему.

Завдяки високій швидкості розробки програмного продукту та меншій вартості розробки, організації, що займаються програмним забезпеченням, зосереджуються на застосуванні гнучких методів у розподілених середовищах, які відомі як розподілена розробка програмного забезпечення (DSD). Це дослідження включає розподілені гнучкі практики та зосереджено на визначенні передбачуваних викликів та їхніх пріоритетів у контексті гнучкого розвитку в розподіленому середовищі. Був проведений огляд літератури, щоб з’ясувати ці проблеми, і вони були поділені на чотири категорії, такі як:

- команда,
- процес,
- сучасна технологія та
- менеджмент.

Виконано порівняння завдань двох досліджень, а також визначення пріоритетності завдань за основними чотирма категоріями.

2.2 Інженерні виклики вимог для великомасштабних гнучких проектів

Використання гнучких стратегій популярне в індустрії програмного забезпечення завдяки численним перевагам. У складних проектах розробки програмного забезпечення, таких як управління ланцюгом поставок, які мають нестабільні та невизначені вимоги, розробка ускладнюється, якщо

використовується передбачувана модель процесу. Оскільки гнучкі стратегії є адаптивними, їх можна використовувати, але використання гнучких методологій допомагає та покращує один аспект великомасштабної розробки програмного забезпечення (LS-SD), створюючи проблеми в іншій частині.

Відомі проблеми, пов’язані з RE, які застосовують гнучкі практики в LS-SD, такі як велика тривалість RE через складний процес прийняття рішень, труднощі у створенні та підтримці списку пріоритетів вимог і час очікування вимог.

У перерахованих проблемах RE бракує високорівневого управління вимогами, наявні проблем уточнення, розрив між короткостроковим і довгостроковим плануванням, а також важко створювати й оцінити user stories. У великих проектах розробки необхідно розуміти та керувати вимогами високого рівня. Отже, обробка вимог високого рівня та їх розуміння є важливими, оскільки добре відомо, що неоднозначність вимог впливає на якість цілого продукту.

Великомасштабні гнучкі проекти використовують модель SCRUM для того, щоб бути більш гнучкими та уникнути надмірної уваги до деталей, для покращення процесів та модернізації процесів їх діяльності. У таких масштабних проектах основною проблемою є масштабування поняття власника продукту, щоб вони могли сприяти необхідному перекладу історій користувачів в пункти специфікації вимог, яких зазвичай багато, включаючи складні взаємозалежності та розподіл цих вимог між командами розробників відносно високого рівня.

У гнучкій розробці процес RE не просто обмежений на початку, як у традиційних моделях, але він продовжується до всього циклу розробки. При постійно мінливих вимогах, технологіях і середовищі існує потреба зосередитися на гнучких методологіях для проектування та розробки програмного забезпечення вчасно з меншою кількістю дефектів і ранніми релізами, таким чином досягаючи потрібного рівня задоволеності клієнтів. При цьому можливі проблеми координації, з якими стикаються у великомасштабних гнучких проектах. Оскільки процес прийняття рішень складний для розробників,

які використовують неформальний спосіб спілкування, проблеми, з якими стикаються учасники, полягають у відсутності взаємодії між учасниками, комунікаційними проблемами, втратою інформації, складними та нестабільними вимогами, складними завданнями взаємозалежності та технічними проблемами.

В [16] автори обговорили виклики інженерії вимог через приклади чотирьох компаній для великомасштабної розробки систем. Серед них двоє – виробники автомобілів, одна – телекомунікаційна компанія і остання – технологічна компанія. У LS-SD коротко обговорюються масштаби гнучкого методу та роль RE. Обговорюються різні випадки компаній, і підсумовується результат кожного випадку.

Основні проблеми розробки вимог гнучкої розробки системи для великого масштабу прикладу 1 пов’язані з комунікацією та управлінням знаннями. Також обговорюються інші виклики, пов’язані з двома сферами знання вимог, такими як спільна цінність користувача та створення та підтримка розуміння системи для прикладу 2. У прикладі 3 основні виявлені проблеми пов’язані зі взаємодією зацікавлених сторін з різних областей. Це домени клієнта, інтеграції, тестування та розробки. У досліджуваному випадку кейс-компаній специфікація вимог потрібна для кращого розуміння поточної системи для аналізу запитів на нові функції та обслуговування системи.

Великі компанії мають ідентичні потреби і потребують відчути ефект змін, перш ніж приймати нові методології у себе. Це пов’язано лише з їхньою складною природою та потребою трансформувати поточні технології та процеси за допомогою існуючих технологій.

Зібраний на основі аналізу досвід потім поширюється між компаніями Nokia, ABB, Daimler Chrysler і Motorola, а також детально обговорюються їхні сфери інтересів. Крім того, на дані 15 пілотних проектів цих чотирьох організацій впливає екстремальне програмування, яке дає ширший огляд впровадження гнучких методів у великих компаніях.

Також обговорюються питання впровадження Agile у великих компаніях. Розробники визначили проблеми з вимогами як потужні драйвери цих чотирьох

організацій. Важко розбити вимоги на детальні специфікації щодо доставки цих вимог командам розробників високого рівня.

2.3 Відомі виклики, їх причини та потенційні рішення при роботі з вимогами в гнучких проектах

Типовими проблемами з програмним забезпеченням можна назвати такі:

1) воно рідко доставляється вчасно через нестабільність графіка, що спричиняє скасування;

2) погане програмне забезпечення , яке не здатне вирішити правильну проблему через те, що вимоги до програмного забезпечення не відповідають потребам і не відповідають вимогам;

3) вигоряння персоналу, коли вони звільняються або коли цікавляться сфери;

4) невеликі зміни вимагають великих зусиль.

Нижче наведено проблеми, які ми виявили в літературі; і їх зведення представлено в таблиці 2.1. Таблиця 2.2 представляє короткий виклад проблем RE, з якими стикаються у великомасштабній гнучкій розробці. Рисунок 2.1 показує короткий зміст цього розділу.

Таблиця 2.1 – Основні проблеми впровадження гнучких технологій при роботі з вимогами

RE виклики	Пропозиції
Пряма комунікація із зацікавленими сторонами	Пряме спілкування, опитування та інтерв'ю
Відсутність документації	Створення деякої документації, слід приділити більше уваги та зосередитися на документації
Відсутні, неоднозначні та суперечливі вимоги	Використовуйте більш формальні способи специфікації вимог, щоб включити відсутні, неоднозначні та суперечливі вимоги

Проблема прототипування	Уникайте багато впровадження під час початкового прототипування.
Недостача попереднього планування, зосередженості та відсутності початкової участі команди	Залучати всіх членів команди проекту з самого початку
Менш досвідчена та кваліфікована команда	Навчіть практикуючого, використовуйте допоміжні інструменти
Невміння клієнтів розповідати історії користувачів	Навчання членів команди і залучення їх до процесу прийняття рішень
Неповне знання про предметну область	Пряма комунікація, спостереження, опитування, інтерв'ю та три чинники: експерт у галузі, процес спілкування та аудиторія допоможуть

Таблиця 2.2 – проблем RE, з якими стикаються у великомасштабній гнучкій розробці

Тип проекту/джерела	Повідомлені проблеми
Різні приклади компаній, які використовують різні гнучкі моделі в проектах, наприклад, банківські, телекомунікаційні та розподілена система	Відсутнє управління вимогами високого рівня, проблеми з уточненням, розрив між короткостроковим і довгостроковим плануванням, важкі створювати та оцінювати історії користувачів
Складні проекти розробки програмного забезпечення, такі як управління ланцюгом поставок, нестабільні та невизначені вимога	Велика тривалість RE через складний процес прийняття рішень , труднощі у створенні та підтримці вимог список пріоритетів
Компанія, яка розробляє вбудовані системи	Повідомляються про проблеми, пов'язані з використанням гнучких практик для великомасштабних проектів програмного забезпечення: планування, слабкі вимоги на початку, слабке визначення пріоритетів вимог і оцінка зусиль, питання якості, роль посередника клієнта, впровадження інновацій, забезпечення РЕ, валідація та верифікація, мотивація до виконання вимог, відсутність документації вимог
Широкомасштабний проект на основі моделі SCRUM з метою	Масштабуйте власників продуктів таким чином, щоб вони могли сприяти необхідному

Тип проекту/джерела	Повідомлені проблеми
модернізації існуючої застарілої системи	«перекладу» історій користувачів, які зазвичай численні, включаючи складні взаємозалежності, і команди розробників відносно високого рівня
Обговорення проблеми координації, з якими стикаються у великомасштабних гнучких проектах	Проблеми, про які повідомляється, полягають у відсутності координації щодо прийняття рішень, комунікації та режиму контролю

Менш прямий зв’язок з замовником викликає проблему відстеження вимог. Це може статися через багато факторів, таких як брак часу, фактор відстані, менша кількість представників клієнтів тощо. З точки зору бізнесу, залучення клієнта та доступність безпосередньо впливають на зміну та підтвердження вимог. У процесі прийняття рішень жодна участь користувача не спричиняє пріоритетності вимог. Відсутня високоякісна взаємодія з користувачем призводить до неправильних вимог.

Щоб вирішити цю проблему, слід проводити пряме спілкування, опитування та інтерв’ю. Збираючи вимоги від клієнта, не зосереджуйтеся на зборі довгострокових вимог і не проводьте довгі офіційні сесії співбесід. Широко використовуйте опитування, щоб отримати відгук від клієнтів.

Відсутність належної документації може створити деякі проблеми для команди розробників і спричинити проблему відстеження вимог. Мінімальна документація в процесі збору вимог створює такі проблеми, як відсутність допомоги персоналу та не забезпечує підтримки методів зворотного проектування. Мінімальні специфікації спричиняють багато проблем, таких як труднощі для команди із забезпечення якості та брак знань про предмет або продукт. Це має більший вплив на великі проекти.

Для усунення варто підготувати певну документацію та приділити більше уваги саме документації, оскільки це допоможе відстежувати, контролювати та перевіряти вимоги. Отже, існує потреба в стандартизації гнучкого процесу та RE діяльності.

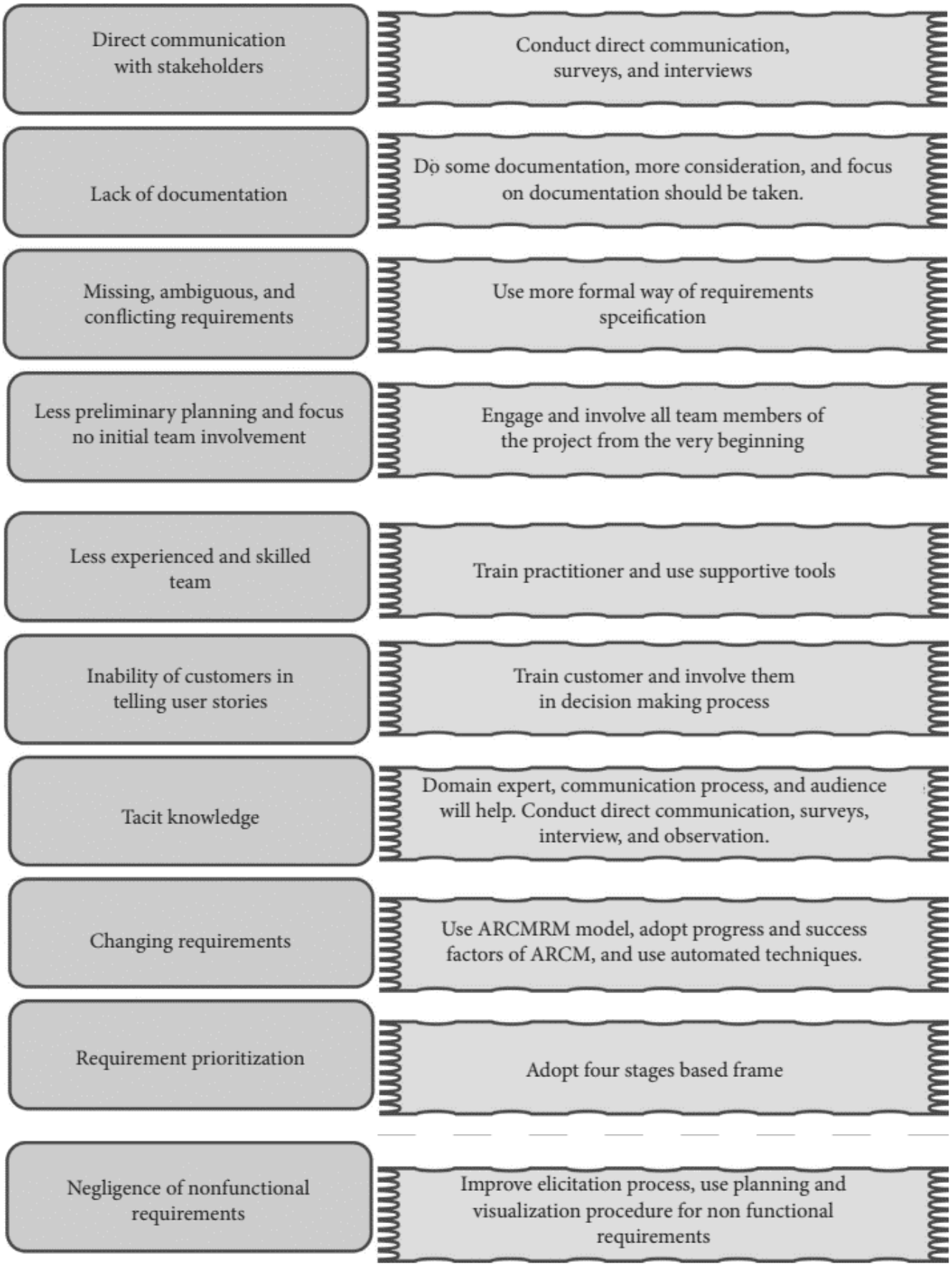

Рисунок 2.1 – Покращення та пропозиції щодо вирішення проблем, що виникли

Відсутні, неоднозначні та суперечливі вимоги. Різні рівні абстракції та непослідовність історій користувачів спричиняють проблеми. Таким чином, вплив неоднозначних вимог на якість і графік є дуже серйозним. Відсутній інтерфейс між вимогами викликає додаткові переробки в наступній ітерації.

Для усунення необхідно використовувати більш формальні способи специфікації вимог, щоб включити відсутні, неоднозначні та суперечливі вимоги

Гнучка розробка RE та використання прототипування створюють у стейкхолдерів неправильне розуміння швидкості розробки. Забагато впровадження в початковому прототипуванні викликає проблеми, такі як проблеми з якістю, породжені практикою повторного використання коду в прототипі, і зацікавлені сторони можуть не прийняти інші цикли розробки, які є більш масштабованими та надійними.

Для вирішення варто уникати великої кількості впровадження на початковому етапі створення прототипу. Паперове прототипування в цьому випадку допоможе багатьма способами, наприклад, діє як перевірка дизайну перед кодуванням, швидко змінюється та усуває специфічні змінні технології.

Менше попереднього планування, зосередженості та відсутності початкової участі команди. На початку розробки RE з меншим попереднім плануванням і фокусом може призвести до повторного виконання вже зробленої роботи. Крім того, відсутність обсягу вимог на початку викликає труднощі в оцінці графіка та бюджету проекту.

Краще й корисно залучати та залучати всіх членів команди проекту з самого початку, щоб вони з самого початку знали все про специфікацію проекту, зміни та технічні деталі. Одна важлива причина залучати та зосереджуватися на всіх членах команд полягає в тому, що оскільки існує мінімум документації, їхнє попереднє планування, зосередженість і залучення є дуже корисними. Обов’язком керівника проекту є контролювати присутність членів команди подумки, залучаючи їх до різноманітних видів діяльності під час анкет та співбесід.

Менш досвідчена та кваліфікована команда. Неправильне тлумачення вимог, неправильне спілкування та неадекватний аналіз вимог є результатом діяльності менш досвідченої та кваліфікованої команди, що, у свою чергу, спричиняє низький рівень успіху проекту та інші проблеми. Особи/команда, залучені до діяльності, створюють неправильні програмні продукти через менший досвід і брак знань.

Як мати справу . Навчіть практиків вдосконалювати свої навички, надаючи їм матеріали, навчальні заняття та вказівки щодо того, як виконувати певну справу. План навчання застосовується до практиків, які працюють у мексиканському SDO із згрупованим набором навичок.

Нездатність клієнтів сформулювати історії користувачів. Нездатність замовника під час опису історій користувачів спричиняє переробку та створює навантаження та додаткові витрати через неповне знання домену серед груп користувачів і, отже, створює проблеми з пріоритезацією вимог. Щоб подолати цю проблему, ми повинні навчити клієнта та залучити його до процесу прийняття рішень протягом усього процесу розробки вимог.

Знання, передані бізнесом команді розробників, є проблемою, оскільки ця інформація не вказана в специфікації. Неявне знання – це те, що ми знаємо, але не можемо сказати; це те, що відоме і набуте на особистому досвіді, тому інколи їх важко сформулювати. У неявних знаннях існує ризик, пов’язаний з нерозумінням вимог і передбаченням, і дуже важко залучити зацікавлених осіб з усіх культур, сфер і професій.

Пряме спілкування, спостереження, опитування та інтерв’ю часто зменшують ймовірність неявного знання. Чинники, такі як експерт предметної області, комунікаційний процес і аудиторія, допоможуть зіткнутися з проблемою неявного знання. Доцільніше розбити зустріч на менші сесії, оскільки це допоможе учасникам засвоїти знання між сесіями та почуватиметься більш комфортно. Одержувач знань (аудиторія на засіданні) повинен мати хороші навички слухання. Увімкніть практику обміну знаннями як позитивного, так і

негативного досвіду, який допомагає членам команди, і вони навчаться на цьому, оскільки обмін знаннями є ключовим фактором успіху в розвитку.

Іноді через зміни потреб замовника розроблений продукт у попередній ітерації викликає проблеми сумісності інтерфейсу. Через зміну вимог гнучкий проект стає дорогим; таким чином зросла вартість проекту та вартість його обслуговування. Нездатність керувати змінами вимог спричиняє збій системи.

Для вирішення можна використати модель, яка складається з трьох основних компонентів, таких як рівень зрілості, рівень факторів і рівень оцінки. Ця модель також підтримує глобальну розробку програмного забезпечення для вимірювання та вдосконалення ARCMRM Крім того, варто використовувати автоматизовані методи та інструменти для управління проблемами вимог, наприклад, JIRA.

Пріоритезація вимог у гнучкому підході після кожної ітерації є складною через відсутність будь-якої техніки, оскільки зацікавлені сторони не залучені до процесу прийняття рішень, а клієнт не завжди доступний для щоденних зустрічей. У цей період кожної ітерації на проекті може відбутись зміна ринкового попиту або зміна вимог через особисті причини. Ці фактори впливають на процес визначення пріоритетів вимог. Існує багато невидимих і неспостережуваних факторів у пріоритезації вимог, і вони мають великий вплив на результат пріоритезації.

Структура встановлення пріоритетів вимог складається з чотирьох етапів: ідентифікація, верифікація, оцінка та встановлення пріоритетів. Представлена модель також допоможе впоратися зі змінами на будь-якому етапі процесу розробки програмного забезпечення.

Недбалість щодо нефункціональних вимог пов’язана з відсутністю чітких правил чи техніки для збору та оцінки нефункціональних вимог (NFR), наприклад, гнучкі методи, екстремальне програмування та SCRUM популярні, оскільки вони забезпечують якісні функціональні вимоги. Якість продукту, що розробляється, перевіряється за відгуками користувачів після завершення кожної

ітерації. Нехтування нефункціональними вимогами затримує виконання проекту, а також глибоко впливає на якість усього продукту.

3 ЗАГАЛЬНІ РЕКОМЕНДАЦІЇ ЩОДО ВДОСКОНАЛЕННЯ ПРОЦЕСУ AGILE-РОЗРОБКИ ВИМОГ

Багато пропозицій та методів для роботи з вимогами в гнучких проектах представлені багатьма дослідниками. Загалом це гнучкі методи впровадження, гнучкі принципи, принципи роботи з вимогами для корпоративних або великих розподілених проектів та інші [17, 18]. Деякі з них проводять аналіз і перераховують фактори успіху. На малюнку 3.1 показано деякі узагальнені пропозиції щодо вдосконалення процесу розробки вимог, що допоможе включити виклики гнучкої розробки.

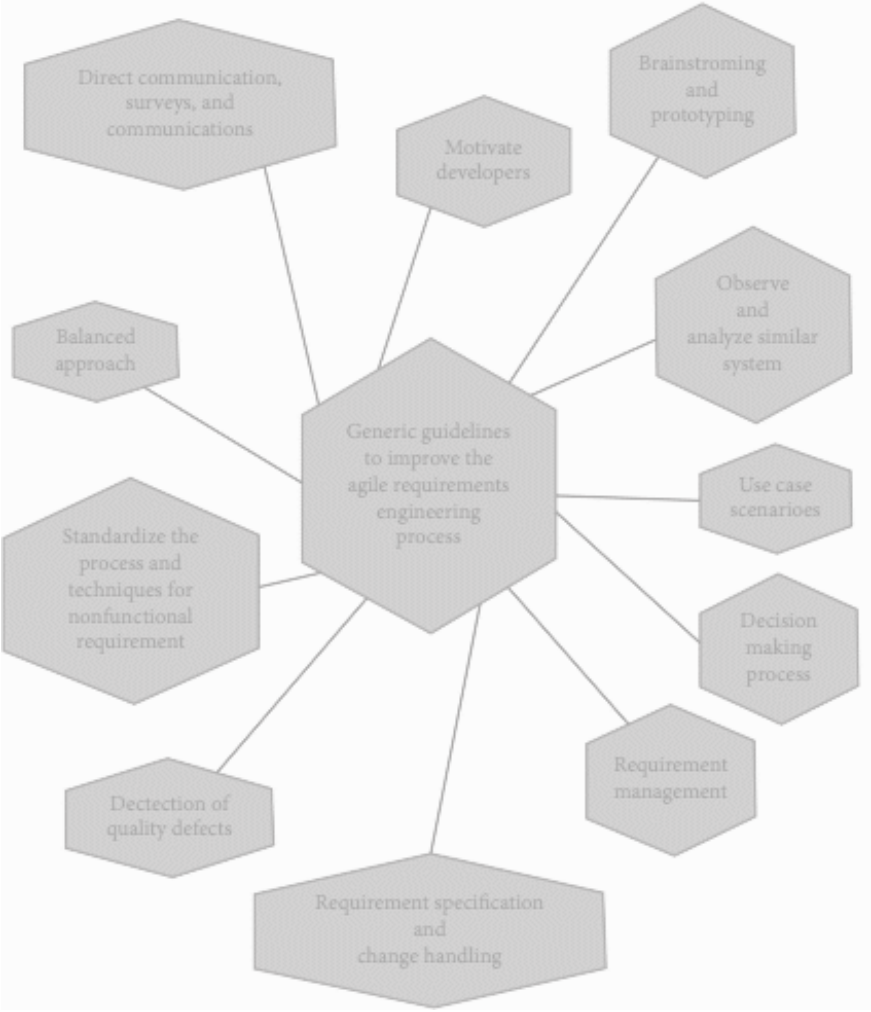

Рисунок 3.1 – Загальні рекомендації стосовно опрацювання вимог в Agile-проектах

Проаналізуємо коротко кожну з цих рекомендацій.

Пряме спілкування, опитування та інтерв'ю. Збираючи вимоги від клієнта, не зосереджуйтеся на зборі довгострокових вимог і не проводите тривалі офіційні сесії співбесід. Широко використовуйте опитування, щоб отримати відгук від клієнтів. Часте спілкування віч-на-віч з самого початку для визначення відповідності вимогам є необхідним і запобігає перевиконанню витрат і графіку проекту. І додатково, коли ми обираємо та збільшуємо залежність від лінійного середовища зв'язку, відповідно рівень дефектів програмного забезпечення також зросте. Отже, необхідно приділяти більше уваги вибору комунікаційного середовища, коли зацікавлені сторони частково доступні.

Розробники мають бути мотивовані. Збирайте та діліться позитивними та успішними історіями та досвідом, щоб мотивувати розробників. Мотивація вважається важливим фактором для продуктивності розробників, і її дуже важко оцінити кількісно. Високовмотивовані розробники більш продуктивні, а отже, у них більше шансів на успіх проекту, і навпаки. Отже, можна зробити висновок, що результат проекту корелює з мотивацією. Таким чином, це відповідальність керівника проекту мотивувати розробників, висловлюючи їм заслуги, вдячність, цінність і побажання.

Мозковий штурм і прототипування. Щоб задовольнити питання змінних вимог замовника, використовуються методи індивідуального та спільного мозкового штурму. Використання прототипування може вирішити багато проблем RE як однозначної документації шляхом розробки виконуваних прототипів, забезпечує мотивацію здійснювати RE, оскільки оновлення прототипу є легким і покращує комунікацію між зацікавленими сторонами, оскільки воно служить візуальним засобом і може фіксувати реальні потреби та результати.

Спостерігайте та аналізуйте подібні системи. Щоб впоратися зі зміною вимог, спостерігайте та аналізуйте вимоги схожих продуктів, які вже створені, тому що іноді не вистачає часу на мозковий штурм, збір і аналіз вимог. У галузі розробки програмного забезпечення проекти певною мірою пов'язані один з

одним за своєю природою або функціями. Навички спостереження та аналізу допоможуть у багатьох відношеннях. Якщо ви вже випробували та розробили деякі подібні системи, то у вас є більше можливостей для вирішення проблем, наприклад повторного використання вже створених компонентів, повторного використання вимог і багатьох інших. Спостереження та аналіз подібних систем покращить ідеї та навички людей.

Сценарії використання. Гнучка структура та методології використовують простий спосіб збору вимог і специфікації у формі історій користувача (user stories), які легко читати, але важко реалізувати, коли ми говоримо про візуальне представлення та розуміння системи. Незважаючи на те, що це дуже тривалий процес, у гнучкому режимі може бути дуже ефективним використання історій користувачів і сценаріїв використання для повнішого охоплення вимог, якщо клієнт легко доступний засобами комунікації.

Процес прийняття рішень. Удосконалюйте процес прийняття рішень і залучайте зацікавлених сторін до процесу прийняття рішень, щоб уникнути проблем із визначенням пріоритетів вимог. Використовуйте модель пріоритезації вимог і методи в Agile для кращого розуміння та для простоти використання, які масштабуються, економлять час і налаштовуються.

правління вимогами. Щоб забезпечити цілісність інформації від користувача (замовника) або вимог у гнучкій системі, повинен існувати відповідний механізм або слід створювати деяку кількість документації, оскільки в гнучкій системі немає уваги до документації та специфікацій. Слід використовувати модель процесу для управління змінами вимог, оскільки вона забезпечує кращий зв’язок, розуміння користувачами, вдосконалення процесу та управління. Слід використовувати метрики управління вимогами [19], а також використовувати інструментальну підтримку для великомасштабних гнучких проектів. Для обробки та структурування вимог слід використовувати таксономії сервіс-орієнтованих засобів (наприклад, Jira).

Специфікація вимог і обробка змін. Використовуйте більш формальні способи специфікації вимог, щоб включити змінні вимоги. Наприклад, JIRA

допомагає справлятися з проблемами гнучких вимог і є більш корисним інструментом для складних проектів.

Процес гнучкого управління змінами вимог складається з ідентифікації змін вимог, аналізу, вартості та оцінки зусиль для обробки та управління змінами вимог. Структура для встановлення пріоритетів вимог у гнучкій розробці може бути представлена у вигляді послідовності виконання чотирьох етапів: ідентифікація, перевірка, оцінка та встановлення пріоритетів. Представлена модель також допоможе впоратися зі змінами на будь-якому етапі процесу розробки програмного забезпечення.

Стандартизуйте процес і методи для нефункціональних вимог. Існує потреба стандартизувати гнучкий процес і RE-діяльність, а також забезпечити спосіб виявлення та керування нефункціональними вимогами, щоб покращити загальну якість продукту на кожній ітерації.

Оприлюднене рішення для вирішення типових проблем, які виникають у великомасштабному проекті, полягає в створенні балансу між гнучким і традиційним підходом. У великих і складних розподілених проектах гнучкість без структури може спричинити хаос, особливо там, де планування, контроль і командна робота є вирішальними. Структура без гнучких концепцій може призвести до жорсткості, особливо коли проект потребує значного навчання, інновацій та змін. Відповідно, документація в гнучкій розробці важлива для підтримки комунікації із зацікавленими сторонами, і вона має більше значення в розподіленій розробці. Тому гнучкі методи з нульовою документацією не рекомендуються. У великих гнучких проектах із зростанням масштабу проектів потрібні більш формальні підходи, такі як централізація, вертикальна комунікація, безособова комунікація та формальний контроль, щоб усунути проблеми, пов’язані з координацією.

Загальними викликами, пов’язаними з гнучкою RE діяльністю, є виявлення вимог, роз’яснення, аналіз, пріоритезація, документування та прийняття рішень. Гнучкий процес розробки програмного забезпечення також інтегрує та забезпечує безпеку програмного забезпечення.

Незважаючи на те, що гнучкий підхід є попереднім традиційним підходам до розробки програмного забезпечення та має численні переваги, він усе ще володіє деякими недоліками, які висвітлюються в цьому розділі, як показано нижче:

- Відсутність повторного використання артефактів.
- Відсутність підтримки розподіленого середовища.
- Не підходить для розробки критичних систем безпеки.

Розробка великих проектів із впровадженням гнучкого підходу складніша. Аналіз гнучкої розробки вимог щодо різних гнучких фреймворків або методологій не обговорюється всебічно. Ми не стверджуємо, що представлене дослідження є універсальним для всіх етапів розробки та методологій. Наш вибір обмежений з точки зору гнучкої розробки вимог, її методологій, проблем і загальних пропозицій.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Методи та засоби психофізіологічного розвантаження як допоміжний процес в розробці ПЗ

Для галузі ІТ інформаційна культура є необхідною умовою виживання, тому що зміна технологій в розробці програмного забезпечення відбувається кожні 6-8 місяців, а інвестиції на підготовку персоналу і освоєння нової технології величезні і великих компаніях варіюються від 1,5 до 2 млрд. доларів на рік [20].

Аналіз свідчить, що інформатизація та інтеграція комунікаційного простору України сприяє різкому підвищенню інформаційної та професійної компетентності, ділової активності, стимулюванню конкуренції, створенню інноваційних підприємств та організацій, нових робочих місць, зниженню витрат на утримання управлінського апарату [20]. Поряд із задачами і здобутками окреслилися негативи використання інформаційних технологій:

1) надмірне інформаційне навантаження, суть якого полягає у тому, що кількість корисної інформації, яка надходить до мережі, перевищує психофізіологічні можливості її сприйняття людиною;

2) велика кількість інформації, яка сприймається, але не є корисною для фахівців в даний момент;

3) інформаційний голод, причиною якого є саме надлишок інформації, викликаний інформаційним перенавантаженням;

4) «інформоманія» як хвороба людини, яка робить останню знеособленою, залежною від перебування в інформаційному просторі і роботи з комп’ютером і чому вона віддає перевагу, уникаючи «живого» спілкування з людьми;

5) поява «кіберспільнот», що за своїми соціокультурними характеристиками набагато ближчі до представників інших культур у

глобальному інформаційному просторі, ніж до своєї етнонаціональної спільноти чи решти населення, не охопленого Інтернетом;

6) індивідуалізм і дегуманізація способу життя «мешканців» Інтернету – відсутність готовності ділитися своїми знаннями.

Слід розуміти, що комп'ютерні технології істотно впливають на життєдіяльність людини, припускаючи глобалізацію і технократизацію суспільства. Але в ще більшій мірі цей вплив поширюється безпосередньо на центральну нервову систему, яка звикає працювати в дуже інтенсивному режимі багатозадачності, де вже переважають не тривалі логічні роздуми, а інтуїтивно-реактивні ланцюжки розумових формулювань у зв'язку з величезним обсягом оброблюваної щодня інформації, кількість якої зростає за експоненціальною швидкістю. Виникає припущення, що саме збільшення обсягу інформації та прискорення її обробки людиною може згубно вплинути на розвиток розумових здібностей людини.

У [21] наведено перелік протипоказань з боку органів зору та загальних (соматичних) протипоказань, які забороняють роботу на ЕОМ, а також комплекс вправ для поліпшення здоров’я і підвищення працездатності.

Таким чином, за умови високого рівня робіт з ЕОМ рекомендується психофізіологічне розвантаження у спеціально обладнаних приміщеннях (кімнати психофізіологічного розвантаження) під час регламентованих перерв або в кінці робочого дня.

При проведенні сеансів психофізіологічного розвантаження рекомендується використовувати деякі елементи методу аутогенного тренування, який ґрунтується на свідомому застосуванні комплексу взаємопов’язаних прийомів психічної саморегуляції й виконанні нескладних фізичних вправ зі словесним самонавіюванням.

У рекомендованому сеансі, який має проводитися у кімнаті психофізіологічного розвантаження з відповідним інтер'єром та кольоровим оформленням, виділяють такі три періоди, або фази:

Перший – абстрагування працівників від виробничої обстановки – відповідає фазі залишкового збудження. Звучить повільна мелодійна музика, пташиний спів. Обравши зручну позу, працівники адаптуються і психологічно готуються до наступних періодів.

Другий – заспокоєння – відповідає фазі відновлювального гальмування. Пропонується показ фото слайдів із зображеннями квітучого лугу, березового гаю, гладенької поверхні ставка тощо. Через навушники транслюється спокійна музика.

Як функціональне освітлення застосовують зелене світло. Яскравість світла має поступово знижуватися впродовж періоду заспокоєння, а наприкінці його світло вимикається зовсім на 1– 2 хв. Екран теж гасне.

Третій – активізація – відповідає фазі підвищеної збудженості.

На початку періоду світло вимкнене, через певний час на екрані з'являється червона пляма, розміри й яскравість якої поступово збільшуються.

Наприкінці періоду звучить бадьора музика. Вимовляються тричі мобілізуючі формули аутогенного тренування, яким мають передувати глибокий вдих та довгий глибокий видих.

Після сеансів психофізіологічного розвантаження у працівників зменшується відчуття втоми, з'являється бадьорість, хороший настрій. Загальний стан відчутно поліпшується.

4.2 Попередження аварій на виробництвах із застосуванням хлору

Хлор є частиною таблиці хімічних елементів і розташовується в ній під номером 17. У природі він зустрічається виключно у формі газу. Найчастіше він має специфічний зелений з жовтим переливом колір. Цей елемент важчий за повітря в 2,5 рази, тому накопичується в підвалах будинків, а на пересіченій місцевості в ярах і низинах. У воді ж хлор розчиняється без сліду і його наявність помітно тільки при великій концентрації (за рахунок специфічного запаху) [22].

В організмі людини в середньому міститься 95 г хлору. За добу людина споживає 5-10 г хлору (кухонна сіль). Він потрібен для вироблення в шлунку соляної кислоти, яка сприяє травленню і знищенню хвороботворних бактерій. Добова потреба хлору для людини становить 800 мг.

Хлор широко застосовується на виробництві, на його основі виготовляють отрутохімікати, розчинники, засоби для дезінфекції та миття, медикаменти. Хлор використовується в кольоровій металургії, у виготовленні пластмас тощо. Також хлор з успіхом застосовується і в побуті для очищення, відбілювання, прання. Завдяки незначним витратам і досить високої ефективності дезінфекції, хлор активно використовується для очищення і знезараження води в плавальних басейнах і питної водопровідної води.

Отруєння хлором можливе в разі:

– перевищення максимально допустимих концентрацій хлору для знезараження води в трубопроводі (сильний запах хлору);

– наявність хлору у великій кількості у воді басейну і часте купання в ньому;

– відбілювання і прання в закритому не провітрюваному приміщенні;

– аварії на підприємстві;

– використання хлору в якості зброї масового ураження.

В організм хлор потрапляє через слизові оболонки дихальної і травної систем, шкіру.

Ознаки отруєння хлором. До перших ознаках отруєння хлором відносяться:

– дискомфорт і подразнення слизової дихальних шляхів;

– підвищене слиновиділення і спазм голосових зв’язок;

– кашель і утруднене дихання;

– відчуття різі та печіння в очах, сльозотеча;

– нудота і гіркота у роті;

– головні болі і можливі судоми.

При попаданні на шкірний покрив або слизові спостерігається значний свербіж і гіперемія (почервоніння), вірогідні підшкірні крововиливи без пошкодження цілісності шкіри.

Тяжкість патологічного процесу та симптоми отруєння хлором знаходяться в прямій залежності від дози отруйної речовини (хлору) і тривалості його дії.

До прибуття медиків слід надати домедичну допомогу потерпілому:

– усунути джерело надходження отрути в організм – вивести або винести потерпілого поза зону дії отруйної речовини. При цьому необхідно пам’ятати про безпеку рятувальника – застосування марлевої маски або респіратора.

– забезпечити доступ чистого повітря;

– зняти забруднений одяг і теплою (не гарячою) водою промити контактуючі ділянки шкіри.

– у разі перорального надходження (проковтування) хлорвмісних рідин, потрібно промити шлунок. Промивати краще через зонд, або можна викликати блювання після рясного пиття.

– у разі пошкодження очей, промивання великою кількістю води або слабким розчином соди для зняття подразнення;

– полоскання ротової порожнини та носа содовими розчинами для мінімізації ушкодження слизових оболонок, застосування інгаляцій з додаванням соди для полегшення кашлю.

До профілактичних заходів отруєння хлором належать:

– забезпечення належних умов праці відповідно до санітарно-технічних вимог (вентиляція, провітрювання, справне обладнання);

– використання індивідуальних засобів захисту при роботі з хімікатами на виробництві;

– регулярні перевірки концентрацій хлору в повітрі робочої зони;

– проведення профілактичних медичних оглядів для виявлення схильності (доклінічних форм) і хронічних захворювань;

– дотримання вимог безпеки у використанні хлорвмісних рідин в побуті.

Суб'єкт господарської діяльності зобов'язаний забезпечити працівників хлорних об'єктів спеціальним одягом, спеціальним взуттям та іншими засобами індивідуального захисту відповідно до Положення про порядок забезпечення працівників спеціальним одягом, спеціальним взуттям та іншими засобами індивідуального захисту:

а) для захисту органів дихання – фільтруючими протигазами, ізолюючими дихальними апаратами та ізолюючими костюмами;

б) для захисту очей – захисними окулярами;

в) для захисту шкіри від їдких речовин – гумовими або прогумованими рукавицями, гумовими чоботами або шкіряними черевиками, сукняними костюмами.

Враховуючи значний ризик і широке застосування хлору, тяжкість ураження і високу можливість летального наслідку, у кожного повинен бути сформований алгоритм дій і чітка позиція – попередити отруєння легше і доцільніше, ніж лікувати і боротися з його наслідками.

ВИСНОВКИ

Розробка вимог є критично важливою фазою життєвого циклу розробки програмного забезпечення. Існують численні виклики, пов’язані з розробкою вимог у процесі гнучкої розробки. Хоча гнучкий процес розробки вимог забезпечує переваги перед традиційними циклами розробки завдяки таким функціям, як мінімум документації, швидкі стратегії впровадження, включення мінливих вимог і часті відгуки від клієнтів, він також має проблеми та обмеження. Ці виклики створюють несприятливий вплив на продукт, що розробляється, так що це впливає на якість і графік розробки проекту. У цій роботі ми визначили проблеми та виклики ER для Agile шляхом огляду останніх сучасних літературних даних.

Сформульовані виклики та пропозиції, засновані на цьому дослідженні, можуть бути використані для вдосконалення навичок і вмінь компаній-розробників під час гнучкої розробки програмного забезпечення. Це дуже корисно для дослідників, які хочуть працювати в цій дослідницькій галузі, яка потребує практичних результатів і досліджень із багатьма підходами Agile, такими як Crystal та екстремальне програмування. Конфлікти та виклики виникають, коли процес не визначений. Таким чином, висновок, зроблений у результаті цього дослідження, полягає в тому, що гнучка сфера все ще незріла, і потреба у встановленні керівних принципів і стандартизації процесу має вирішальне значення для покращення результатів. Таким чином, існує потреба в менеджменті та стандартизації процесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kharchenko, A., Raichev, I., Bodnarchuk, I., & Matsiuk, O. (2021, October). The Survey of Global Software Design Processes. In 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T) (pp. 291-294). IEEE.

2. S. Balaji and M. S. Murugaiyan, “Waterfall vs. V-Model vs. Agile: a comparative study on SDLC,” International Journal of Information Technology and Business Management, vol. 2, no. 1, pp. 26–30, 2012.

3. M. Ochodek and S. Kopczynska, “Perceived importance of˙ agile requirements engineering practices - a survey,” Journal of Systems and Software, vol. 143, pp. 29–43, 2018.

4. X. Wang, L. Zhao, Y. Wang, and J. Sun, “The role of requirements engineering practices in agile development: an empirical study,” in Requirements Engineering, pp. 195–209, Springer, Berlin, Germany, 2014.

5. A. Sillitti and G. Succi, “Requirements engineering for agile methods,” in Engineering and Managing Software Requirements, pp. 309–326, Springer, Berlin, Germany, 2005.

6. Y. Sebega and E. Mnkandla, “Exploring issues in agile requirements engineering in the South African software industry,” The Electronic Journal of Information Systems in Developing Countries, vol. 81, no. 1, pp. 1–18, 2017.

7. H. Hajjdiab and A. S. Al Shaima Taleb, “Adopting agile software development: issues and challenges,” International Journal of Managing Value and Supply Chains, vol. 2, no. 3, pp. 1–10, 2011.

8. Alam, S., Bhatti, S. N., Shah, S. A. A., & Jadi, A. M. (2017). Impact and challenges of requirement engineering in agile methodologies: A systematic review. International Journal of Advanced Computer Science and Applications, 8(4).

9. Bano, M., Imtiaz, S., Ikram, N., Niazi, M., & Usman, M. (2012). Causes of requirement change-a systematic literature review.

10. Dalpiaz, F., & Brinkkemper, S. (2018, August). Agile requirements engineering with user stories. In 2018 IEEE 26th International Requirements Engineering Conference (RE) (pp. 506-507). IEEE.

11. Bodnarchuk, I., Lisovyi, V., Kharchenko, O., & Galai, I. (2018, September). Adaptive Method for Assessment and Selection of Software Architecture in Flexible Techniques of Design. In 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 1, pp. 292-297). IEEE.

12. Jørgensen, M. (2018, May). Do agile methods work for large software projects?. In International Conference on Agile Software Development (pp. 179-190). Springer, Cham.

13. Abrar, M. F., Khan, M. S., Ali, S., Ali, U., Majeed, M. F., Ali, A., ... & Rasheed, N. (2019). Motivators for large-scale agile adoption from management perspective: A systematic literature review.

14. Kovitz, B. (2003). Hidden skills that support phased and agile requirements engineering. Requirements Engineering, 8(2), 135-141.

15. Cho, J. J. (2010). An exploratory study on issues and challenges of agile software development with scrum. All Graduate theses and dissertations, 599.

16. Kasauli, R., Liebel, G., Knauss, E., Gopakumar, S., & Kanagwa, B. (2017, September). Requirements engineering challenges in large-scale agile system development. In 2017 IEEE 25th International Requirements Engineering Conference (RE) (pp. 352-361). IEEE.

17. Rolland, K., Dingsoyr, T., Fitzgerald, B., & Stol, K. J. (2016). Problematizing agile in the large: alternative assumptions for large-scale agile development. In 39th International Conference on Information Systems (pp. 1-21). Association for Information Systems (AIS).

18. Dorairaj, S., Noble, J., & Malik, P. (2012, August). Knowledge management in distributed agile software development. In 2012 Agile Conference (pp. 64-73). IEEE.

19. Kumar, S. A., & Kumar, T. A. (2011). Study the impact of requirements management characteristics in global software development projects: an ontology based approach. International Journal of Software Engineering & Applications, 2(4), 107.

20. Пивоваров М.Г., Медко Д.А. Перспективы создания и развития информационно-коммуникационной системы Украины // Економіка: проблеми теорії та практики: Зб. наук. праць. – Вип. 49. –Дніпропетровськ: Дніпропетр. Нац. Ун-т, 2000. – С. 56-61.

21. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин МОЗ України від 10.12.1998 № 7. // Офіційний сайт Верховної Ради України. – [Електронний ресурс]. – Режим доступу https://zakon.rada.gov.ua/rada/show/v0007282-98

22. Бідяк О. Профілактика отруєння хлором. // Офіційний сайт управління держпраці в Тернопільській області. – [Електронний ресурс]. – Режим доступу https://te.dsp.gov.ua/profilaktyka-otruyennya-hlorom/

ДОДАТКИ

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

М А Т Е Р І А Л И

X НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

7–8 грудня 2022 року

ТЕРНОПІЛЬ
2022

УДК 004.41
О. Гузеляк, Ю. Шевчук, Б. Береженко, І. Боднарчук
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ПРОГРАМНА АРХІТЕКТУРА В РОЗПОДІЛЕНИХ КОМАНДАХ ГНУЧКИХ ПРОЄКТІВ

UDC 004.41
O. Huzeliak, Yu. Shevchuk, B. Berezhenko, I. Bodnarchuk

SOFTWARE ARCHITECTURE DEGIGN IN DISTRIBUTED TEAMS OF AGILE PROJECTS

Базуючись на принципах Agile Маніфесту [1] практики Agile розглядають первинну розробку програмної архітектури (ПА), як витрати ресурсів, які можуть не окупитись. В Agile традиційно приділяли основну увагу при плануванні ітерацій потребам замовника і сценаріям використання (так званим user stories), залишаючи поза увагою архітектурне проектування.

Однак, при застосуванні Agile до великих проектів ці погляди змінились. Так в роботі [2] показано, що ігнорування архітектурного проектування може спричинити неоптимальні рішення, а в роботі [3] автор стверджував, шо невідповідність розробки великих проектів архітектурному проекту може привести до їх провалу. В роботах [3], [4] підкреслена ключова роль архітектурного проекту при застосуванні Agile до розробки великих і складних проектів, а також фактичну неможливість їх виконання без використання архітектури. Це відбувається через втрату зв'язку беклогів (специфікацій вимог) з усією системою, що робить неможливим передбачити реальні наслідки своїх рішень при плануванні кожної ітерації.

Для вирішення цих проблем були впроваджені в ітерації гнучких методів елементи архітектурного проектування та почали реалізовувати зв'язок архітектурного проекту всієї програмної системи з проектами локальних команд [6]. При реалізації великих проектів в рамках гнучких методів створювалась «координуюча» команда, яка вела архітектурний проект всієї програмної системи і здійснювала координацію внесення змін в архітектури компонентів, які розробляли локальні команди [6]. Для цього були введені нові ролі в координуючій команді. Це архітектор системи, архітектор бізнес-процесів та інші. Процеси архітектурного проектування в локальній команді може вести або архітектор, або один з членів команди, суміщаючи ці обов'язки з роллю розробника чи іншою роллю на проєкті [7]. Для забезпечення ефективності реалізації взаємодії процесів на локальному і глобальному рівнях необхідно було розробити відповідні моделі цих процесів та створити CASE-засіб, який би автоматизував процеси створення та обміну архітектурною інформацією. В [8] було запропоновано двоетапну схему такої взаємодії (див. рис. 1). На першому етапі створюються альтернативні варіанти архітектури з врахуванням вимог за технологією, яка залежить від прийнятого стилю документування. Для оцінювання альтернатив обчислюються їх відносні оцінки по кожному з критеріїв якості з використанням модифікованого методу аналізу ієрархій (МАІ). Вибір кращого варіанта архітектури виконується методом аналізу компромісів або на основі значення інтегрального показника якості [9].

На другому етапі вибраний варіант архітектури впроваджується в гнучкому методі розробки. В разі виявлення нових вимог до програмної системи (ПС) або зміни існуючих вносяться відповідні зміни в архітектуру. Зміни виконуються шляхом корегування коду відповідного патерну (шаблону) проєктування або його заміною на альтернативний. Для дослідження та оптимізації впливу цих змін на якість проводиться корекція значень критеріїв якості. Процедура корекції розроблена на основі застосування методу «заміщення – компенсації», в якому збільшення деяких критеріїв якості відбувається за рахунок зменшення інших, що дозволяє оптимізувати ці зміни і не виходити за межі бюджету проекту [10].

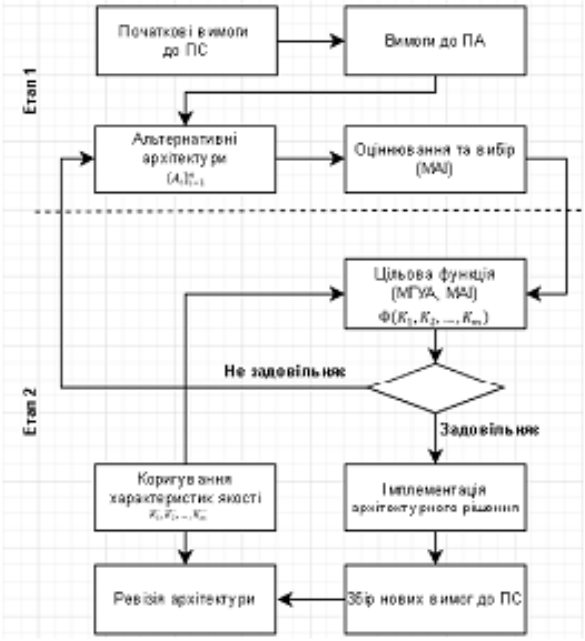

Рисунок 1. Схема проєктування архітектури в розподілених командах для гнучких проєктів

Перший і другий етапи об'єднуються процедурою обчислення цільової функції якості архітектури. Цільова функція визначається методом групового урахування аргументів в поєднанні з MAI [8]. Запропонована процедура дозволить, крім координації процесів, також забезпечити якість проєкту. Для адаптації процесів архітектурного проектування до вимог Agile необхідно максимально зменшити час на процеси архітектурного проектування. Для цього пропонується створити систему, яка б могла підтримувати спільне використання командами архітектурної інформації. Ці процедури повинні виконуватись командами в ітераціях Agile, а також координуючою командою для архітектури всієї системи. Архітектурний проект приймається і створюється на фазі аналізу вимог перед спринтами розробки. Проте в ході розробки може виникнути потреба змінювати архітектуру, і командою SCRUM або архітектором приймаються певні нові рішення. Основною задачею тут є оперативне оцінювання можливих архітектурних рішень та вибору найбільш прийнятної для задоволення вимог всіх зацікавлених сторін. Тут може бути використаний MAI, а для коригування архітектури в ітераціях той же MAI або метод групового урахування аргументів (МГУА), які набагато ефективніші ніж метод сценаріїв ATAM, що розглядався в [14]. Також необхідно автоматизувати процедури формування альтернативних архітектур на основі архітектурно значимих вимог та при внесені змін в програмний код і здійсненні реінжинірингу. Ці процедури, а також метод оцінювання MAI та процедури зберігання архітектурної інформації реалізовані в засобі «Архітектор» [15], [16]. Тому систему можна будувати на його основі.

Отже, хоча Agile підхід має багато переваг він все ж таки не є універсальним рішенням, і компанії часто використовують поєднання гнучких та планово орієнтовних методів, що називається гібридним підходом. Він є результатом компромісу використання переваг ітеративного, логічного та спільного підходів із підтримкою певного рівня планування та структури. При цьому гібридний підхід має ряд переваг, таких як: належна підтримка бізнес-проектів, допомагає досягти хорошої якості продукції, адекватно відповідає зовнішнім вимогам (наприклад, стандартам) та допомагає пришвидшити розробку й скоротити час виходу продукту на ринок, а також дає можливість постійно вдосконалювати якість завдяки гібридним конструкціям, покращити задоволеність команди. Гібридний підхід має також

недоліки: проблеми забезпечення продуктивності проекту та затримки релізів. Мінімізацію недоліків реалізують застосуванням ширшого впровадження архітектурного проектування в Agile, а також застосуванням засобів автоматизації цих процесів.

Література

1. Agile alliance: manifesto for agile software development. URL: http:// agilemanifesto.org [retrieved: 10.10.2022]
2. Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. Information and software technology, 50 (9–10), 833–859.
3. Bowers, J., May, J., Melander, E., Baarman, M., & Ayoob, A. (2002, August). Tailoring XP for large system mission critical software development. In Conference on Extreme Programming and Agile Methods (pp. 100–111). Springer, Berlin, Heidelberg.
4. Ghanam Y. Andreichuk D., Maurer F. Reactive variability management in agile Software development. In: Agile conference. Orlando, FL: Computer Society; 2010, p. 27–34.
5. Abrahamsson, P., Babar, M. A., & Kruchten, P. (2010). Agility and architecture: Can they coexist? IEEE Software, 27 (2), 16–22.
6. Dingsøyr, T., Moe, N. B., & Seim, E. A. (2018). Coordinating knowledge work in multiteam programs: findings from a large-scale agile development program. Project Management Journal, 49 (6), 64–77.
7. Eloranta, V. P., & Koskimies, K. (2013). Software architecture practices in Agile enterprises. In Aligning Enterprise, System, and Software Architectures (p. 230–249). IGI Global.
8. Bodnarchuk, I., Lisovyi, V., Kharchenko, O., & Galai, I. (2018, September). Adaptive Method for Assessment and Selection of Software Architecture in Flexible Techniques of Design. In 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 1, p. 292–297). IEEE.
9. Kharchenko, A., Bodnarchuk, I., Halay, I., & Yatcyshyn, V. (2016). An Optimal Trade-off Solution of the Software Architecture Choice Problem. Journal of Information and Computing Science. 11 (4). P. 281–290.
10. I. Bodnarchuk, et al. «Multicriteria choice of software architecture using dynamic correction of quality attributes.» International Conference on Computer Science, Engineering and Education Applications. Springer, Cham, 2019, p. 419–427.
11. Galster M., & Avgeriou P. (2014). Supporting variability through agility to achieve adaptable architectures. In Agile Software Architecture (p. 139–159). Morgan Kaufmann.
12. KharchenkoA., Halay I., & Bodnarchuk I. (2016, September). Multicriteria architecture choice of software system under design and reengineering. In 2016 XIth International Scientific and Technical Conference Computer Sciences and Information Technologies (CSIT) (p. 4–8). IEEE.
13. Харченко О. Г., Боднарчук,І. О., Райчев І. Е., & Галай І. О. (2015). Інструментальний засіб порівняльного оцінювання і багатокритеріального вибору архітектури програмних систем. Інженерія програмного забезпечення. Київ: НАУ, 2015. № 1. С. 10–24.