

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВАДИМА ГЕТЬМАНА**

**Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»**

Кафедра _системного аналізу та кібербезпеки

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА

«Системний аналіз»

СПЕЦІАЛЬНІСТЬ

124 «Системний аналіз»

Форма навчання: очна (денна)

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

**на тему «Системний підхід до розробки й управління вимогами до програмних
продуктів в ІТ-компанії»**

здобувача Скаржевського Ярослава Олександровича

**Науковий керівник: канд.екон.наук, доцент
Лазарева Світлана Федорівна**

**Робота допущена до захисту перед екзаменаційною комісією
з атестації здобувачів вищої освіти (ЕК)**

**Завідувач кафедри: доктор фізико-математичних наук професор
Джалладова Ірада Агаєвна**

Київ 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ
ВАДИМА ГЕТЬМАНА

Навчально-науковий інститут
«Інститут інформаційних технологій в економіці»

Кафедра системного аналізу та кібербезпеки
(назва кафедри)

ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА
СПЕЦІАЛЬНІСТЬ
СПЕЦІАЛІЗАЦІЯ (пакет)

«Системний аналіз»
124 «Системний аналіз»

ПОГОДЖЕНО

Керівник проектної групи (гарант)
освітньо-професійної програми

_____ Джалладова І. А. _____
(підпис) (ініціали, прізвище)

_____ 20__ р.

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Джалладова І. А. _____
(підпис) (ініціали, прізвище)

_____ 20__ р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

здобувачу вищої освіти Скаржевський Ярослав Олександрович
(прізвище, ім'я, по батькові)

очної форми навчання
очної (денної), заочної, дистанційної

на підготовку кваліфікаційної магістерської роботи
на тему «Системний підхід до розробки й управління вимогами до програмних
продуктів в ІТ-компанії»

Тему затверджено наказом ректора Університету від " _____ " _____ 20__ р. № _____

Кваліфікаційна магістерська робота виконується на матеріалах

План кваліфікаційної магістерської роботи	
Розділ 1	Теоретико-методичні засади розробки та управління вимогами до програмних продуктів (назва розділу)
Розділ 2	Аналіз діяльності з розробки програмних продуктів компанії ТОВ «Платмастер» (назва розділу)
Розділ 3	Пропозиції щодо удосконалення процесів розробки та управління вимогами в ТОВ «Платмастер» (назва розділу)

Об'єкт дослідження: Процес розробки програмного продукту

Предмет дослідження: продукту	Процес розробки та управління вимогами до програмного
Мета кваліфікаційної магістерської роботи:	Формалізація та представлення системного підходу до розробки та управління вимогами до програмних продуктів в ІТ-компанії

Конкретні завдання, які здобувач повинен виконати для досягнення поставленої мети:

У розділі 1	Проаналізувати значення та роль вимог у процесі розробки програмного Дослідити та висвітлити сучасні методи розробки та управління вимогами. Дослідити та представити основні принципи системного підходу в контексті управління вимогами.
--------------------	--

У розділі 2	Надати загальну характеристику ІТ-компанії ТОВ «Платмастер». Проаналізувати систему проєктного управління, а також процеси розробки та управління вимогами в ТОВ «Платмастер». Змодельовати процеси розробки та управління вимогами в ТОВ «Платмастер».
--------------------	--

У розділі 3	Розробити та запропонувати рішення щодо покращення процесів розробки та управління вимогами. Представити графічну «to be» модель впливу змін на процеси розробки та управління. Надати пропозиції щодо інтеграції процесів роботи з вимогами у систему проєктного управління.
--------------------	--

**Завдання підготував
науковий керівник**

(підпис) Лазарєва С. Ф.
(ініціали, прізвище)

«____» _____ 20__ р.

**Завдання одержав
здобувач**

(підпис) Скаржевський Я. О.
(ініціали, прізвище)

«____» _____ 20__ р.

Реферат

Кваліфікаційна магістерська робота містить 92 сторінки, 14 рисунків, перелік джерел посилання з 50 найменувань, додатки А, Б, В.

«Системний підхід до розробки й управління вимогами до програмних продуктів в ІТ-компанії»

(назва кваліфікаційної магістерської роботи)

Об'єктом дослідження є процес розробки програмного продукту.

Предметом дослідження є процес розробки та управління вимогами до програмного продукту.

Мета кваліфікаційної магістерської роботи – формалізації та представленні системного підходу до розробки та управління вимогами до програмних продуктів в ІТ-компанії.

Відповідно до поставленої мети були визначені такі завдання:

- роз'яснення ролі вимог у процесі розробки програмного продукту;
- огляд сучасних методів роботи з вимогами;
- опис основних принципів системного підходу до управління вимогами;
- аналіз процесів розробки та управління вимогами на прикладі ТОВ «Платмастер»;
- аналіз структури проєктного управління ТОВ «Платмастер»;
- створення графічних моделей процесів розробки та управління вимогами в ТОВ «Платмастер»;
- розроблення пропозицій по вдосконаленню процесів розробки та управління вимогами в ТОВ «Платмастер»;
- розроблення пропозицій щодо інтеграції процесів роботи з вимогами в систему проєктного управління ТОВ «Платмастер».

Практичне значення отриманих результатів. На основі описаних моделей процесів розробки та управління вимогами ІТ-компанії «Платмастер», було надано оцінку їх ефективності. Визначено перспективи та шляхи покращення цих процесів для підвищення загальної продуктивності процесу розробки та якості продуктів компанії. Описані та застосовані прийоми розробки та управління вимогами можуть бути використані на практиці для оцінки та покращення процесів в інших ІТ-компаніях, які використовують методологію Scrum або інші методи гнучкого підходу до розробки.

Рік виконання кваліфікаційної магістерської роботи 2024.Рік захисту роботи 2024.

Ключові слова: розробка вимог, управління вимогами, проєктне управління, системний підхід, Scrum, гнучка розробка, рекомендовані прийоми, виявлення, аналіз, документування, узгодження, замовник, зацікавлені сторони, проєктна команда, проєктний менеджер, аналітик вимог, користувацькі вимоги, системні вимоги, бізнес-вимоги, бізнес-правила, функціональні вимоги, нефункціональні вимоги, Confluence, Jira, Slack

В і д г у к
про кваліфікаційну магістерську роботу
здобувача навчально-наукового інституту
«Інститут інформаційних технологій в економіці»
освітньо-професійної програми « Системний аналіз»
Скаржевського Ярослава Олександровича
на тему: «Системний підхід до розробки й управління вимогами до
програмних продуктів в ІТ-компанії»

1. Актуальність теми: Тема кваліфікаційної магістерської роботи є актуальною, оскільки присвячена дослідженню одного з найбільш важливих процесів розробки програмних продуктів (ПП), що визначають його якість - розробці та управлінню вимогами, які є орієнтиром для проєктної команди та основною ланкою для співпраці із замовником.

2. Позитивні риси кваліфікаційної роботи: В основу КМР покладено системний підхід до розв'язання проблем, пов'язаних з розробкою й управлінням вимогами до ПП, надано пропозиції щодо удосконалення цих процесів в компанії ТОВ «Платмастер», на прикладі якої написана робота.

3. Наявність самостійних розробок автора: Надані рекомендації базуються на результатах дослідження предметної області, виконаного автором самостійно. За результатами дослідження автор розробив удосконалену модель процесів розробки та управління вимогами до програмних продуктів, запропонував шляхи покращення системи проєктного управління в ТОВ «Платмастер».

4. Цінність теоретичних висновків та практичних рекомендацій: Теоретичні висновки є коректними, базуються на опрацюванні значної кількості першоджерел. Це дозволило надати слушні пропозиції щодо удосконалення процесів розробки та управління вимогами до програмних продуктів в ТОВ «Платмастер». Автор апробував їх при виконанні реального проєкту в ролі аналітика вимог.

Цінність роботи полягає також у тому, що зроблені в ході дослідження, теоретичні узагальнення, висновки та рекомендації можуть бути використані іншими підприємствами.

5. Наявність недоліків: Недоліків, які могли б вплинути на позитивну оцінку кваліфікаційної роботи не виявлено. Незначні редакційні та стилістичні огріхи не впливають на позитивне враження від роботи

6. Загальна оцінка кваліфікаційної магістерської роботи та її допущення до захисту перед ЕК: При написанні кваліфікаційної бакалаврської роботи Скаржевський Ярослав Олександрович показав себе як сформований фахівець, здатний розв'язувати складні як теоретичні, так і практичні проблеми системного аналізу в ІТ-проєктах. Результати проведеного дослідження викладено в роботі професійно.

Кваліфікаційна магістерська робота Скаржевського Я.О. може бути рекомендована до захисту в ЕК. Робота відповідає вимогам, що висуваються до кваліфікаційних робіт на здобуття освітнього ступеня магістр, заслуговує високої оцінки, а її автор, Скаржевський Ярослав Олександрович, заслуговує на присвоєння йому освітнього ступеня магістр зі спеціальності 124 «Системний аналіз», галузі знань 12 «Інформаційні технології».

Науковий керівник
професор кафедри, канд.екон наук, доцент

_____ Лазарева С.Ф.

24 травня 2024 р.

Рецензія
на кваліфікаційну магістерську роботу
здобувача вищої освіти
Скаржевського Ярослава Олександровича
на тему: «Системний підхід до розробки й управління вимогами до програмних
продуктів в ІТ-компанії»

Актуальність теми кваліфікаційної роботи і доцільність її розроблення

Кваліфікаційна магістерська робота Скаржевського Ярослава Олександровича присвячена важливій темі управління вимогами до програмних продуктів. В умовах сучасного ринку ІТ-технологій, де вимоги до програмних продуктів постійно змінюються, питання їх ефективної розробки та управління ними набуває особливої актуальності. Це дослідження є важливим, оскільки воно спрямоване на вдосконалення процесів розробки та управління вимогами, що, у свою чергу, може суттєво підвищити якість кінцевого продукту, а також прозорість процесу проєктного управління.

Якість проведеного дослідження

Виконане дослідження відповідає всім вимогам, що висуваються до кваліфікаційних магістерських робіт. В основу КМР покладено системний підхід до розв'язання проблем, що розглядаються, а також узагальнення теоретичного і практичного досвіду розробки та управління вимогами до програмних продуктів. Рекомендації з удосконалення, представлені в роботі, а також розроблені проєктні рішення, ґрунтуються на результатах дослідження предметної області та специфіки роботи ІТ-компанії «Платмастер». Поставлені перед здобувачем завдання були повністю виконані, мета роботи досягнута, а Скаржевський Я.О. продемонстрував високий рівень професійних знань та навичок.

Позитивні риси кваліфікаційної роботи

Позитивним аспектом кваліфікаційної роботи є те, що результати дослідження були впроваджені у діяльність ІТ-компанії ТОВ «Платмастер», на прикладі якої написана робота. За результатами дослідження автором запропоновано впровадження сучасних прийомів розробки та управління вимогами в виробничі процеси, а також інтеграцію цих прийомів в проєктне управління за методологією Scrum. Це дозволило покращити ефективність процесів розробки та управління вимогами, знизити ризики та підвищити ефективність проєктного менеджменту.

Зауваження

В роботі не було виявлено суттєвих недоліків. Вона виконана на високому рівні, а незначні стилістичні помилки не впливають на загальний результат.

Практична значимість висновків і рекомендацій

Практична цінність КМР полягає в тому, що в роботі представлено ефективні прийоми розробки та управління вимогами для ІТ-компанії «Платмастер», а також їх використання в контексті методології Scrum. У роботі запропоновано структуру процесів розробки та управління вимогами, проведений аналіз діяльності ІТ-компанії, та висунуто рекомендації щодо вдосконалення цих процесів. Запропоновані зміни забезпечують підвищення ефективності та прозорості командної роботи та співпраці із зацікавленими сторонами.

Проєктні рішення, запропоновані в КМР, можуть бути використані для покращення процесів управління вимогами в інших ІТ-компаніях, що працюють за аналогічною методологією. Робота Скаржевського Я.О. повністю відповідає вимогам, що висуваються до кваліфікаційних робіт магістерського рівня, може бути рекомендована до захисту та заслуговує на оцінку "відмінно".

ТОВ «Платмастер» директор компанії
Доктор філософії

_____ Бугайов М. В.

ЗМІСТ

ВСТУП	3
РОЗДІЛ 1 ТЕОРЕТИКО-МЕТОДИЧНІ ЗАСАДИ РОЗРОБКИ ТА УПРАВЛІННЯ ВИМОГАМИ ДО ПРОГРАМНИХ ПРОДУКТІВ	5
1.1 Значення та роль вимог у процесі розробки програмного продукту	5
1.2 Методи розробки та управління вимогами	15
1.3 Основні принципи системного підходу до управління вимогами	27
РОЗДІЛ 2 АНАЛІЗ ДІЯЛЬНОСТІ З РОЗРОБКИ ПРОГРАМНИХ ПРОДУКТІВ КОМПАНІЇ ТОВ «ПЛАТМАСТЕР»	34
2.1 Загальна характеристика компанії	34
2.2 Аналіз системи проєктного управління на підприємстві	39
2.3 Аналіз процесів розробки та управління вимогами	45
2.4 Модель процесів розробки та управління вимогами прийнята в компанії ..	50
РОЗДІЛ 3. ПРОПОЗИЦІЇ ЩОДО УДОСКОНАЛЕННЯ ПРОЦЕСІВ РОЗРОБКИ ТА УПРАВЛІННЯ ВИМОГАМИ В ТОВ «ПЛАТМАСТЕР»..	62
3.1 Систематизація та структуризація процесу розробки вимог.....	62
3.2 Систематизація та структуризація процесу управління вимогами	77
3.3 Інтеграція процесів розробки та управління вимогами в систему проєктного управління	82
ВИСНОВКИ.....	85
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ	87
ДОДАТКИ.....	93

ВСТУП

У сучасному IT-середовищі, розробка програмних продуктів є складним та відповідальним процесом, особливо в контексті розробки та управління вимогами. Ця задача є основною функцією аналітика під час впровадження IT-проектів, а результат роботи напряду впливає на швидкість процесу розробки та якість вихідного програмного продукту. Системний підхід до розробки та управління вимогами об'єднує в собі методи, які дозволяють організувати структуровані процеси роботи з вимогами, взаємодії із стейкхолдерами, а також спростити роботу та підвищити компетентність аналітика. Розробка та впровадження такого підходу в компаніях з розробки програмних продуктів може допомогти формалізувати та оптимізувати процес роботи аналітика та всієї проєктної команди. Системний підхід до розробки та управління вимогами потенційно матиме особливий вплив на українські підприємства, оскільки в них ще не сформовані строгі підходи до процесу розробки. Описані методики роботи з вимогами в рамках гнучкої розробки додатково розкривають можливості такого підходу на підприємстві та як гнучкий підхід надає компанії можливість створювати конкурентноспроможний продукт в умовах економічної нестабільності.

У цій роботі будуть розглянуті та досліджені твори зарубіжних авторів, такі як наукові публікації, інтернет джерела. В яких висвітлено методи та підходи до розробки та управління вимогами до програмного продукту, поняття, принципи та кращі практики які використовуються аналітиками вимог в компаніях по всьому світу.

Мета даної КМР полягає у формалізації та представленні системного підходу до розробки та управління вимогами до програмних продуктів в IT-компанії.

Для досягнення цієї мети в роботі поставлені такі завдання:

- дослідження ролі та місця процесу формування вимог в життєвому циклі програмних продуктів (ПП);

- дослідження сучасних методів розробки та управління вимогами в контексті розробки ПП;
- дослідження принципів системного підходу в контексті роботи з вимогами
- аналіз діяльності, а також структури та процесів проєктного управління та процесів розробки та управління вимогами в ІТ-компанії ТОВ «Платмастер»;
- розробити графічні моделі as is діяльності та бізнес-процесів розробки та управління вимогами до ПП на підприємстві;
- за результатами аналізу надати пропозиції щодо удосконалення цих процесів із графічним представленням результатів впливу пропонованих змін (модель to be) та інтеграції в систему проєктного управління компанії.

Об'єктом дослідження є процес розробки програмного продукту.

Предметом дослідження є процеси розробки та управління вимогами до програмного продукту.

Під час написання роботи були використані такі методи: спостереження, формалізація, узагальнення, аналіз, співставлення, графічний метод, а також емпіричні методи дослідження.

Було проведено спостереження за процесом роботи проєктної команди в ІТ-компанії «Платмастер». За допомогою співставлення було досліджено вплив змін у проєктному управлінні на процес роботи над вимогами. Індуктивний та дедуктивний методи розкривають стан проєкту та програмного продукту на час його розробки. Аналіз загальної структури процесу розробки висвітлює переваги та недоліки певних управлінських та оперативних рішень членів команди.

РОЗДІЛ 1

ТЕОРЕТИКО-МЕТОДИЧНІ ЗАСАДИ РОЗРОБКИ ТА УПРАВЛІННЯ ВИМОГАМИ ДО ПРОГРАМНИХ ПРОДУКТІВ

1.1 Значення та роль вимог у процесі розробки програмного продукту

Вимоги відіграють критично важливу роль у всьому життєвому циклі програмного продукту. Вони визначають, що саме має бути розроблено, яким чином продукт повинен функціонувати та які обмеження та критерії якості необхідно врахувати.

Проте, незважаючи на їх важливість, немає однозначного визначення, терміну "вимоги". У різних контекстах і методологіях під вимогами можуть розуміти різні аспекти – від загальних бізнес-цілей до конкретних технічних специфікацій. Ця неоднозначність ускладнює процес документування та управління вимогами, особливо в умовах динамічних проєктних середовищ, де зміни є частиною нормального робочого процесу.

Розглянемо яку все ж роль грають вимоги до програмного продукту в рамках проєкту його розробки. Консультант Брайан Лоренс припустив, що вимоги – це «щось таке, що визначає вибір дизайну», таке неформальне визначення хоч і є правдивим, проте не розкриває цієї суті вимог і навіть дає хибне враження того що вимоги це виключно щось що визначає дизайн, тобто візуальне представлення продукту.

Визначення вимог сформульоване Іаном Сомервіллем та Пітом Сойером у їх творі «Requirements management: A good practice guide» звучить наступним чином - вимоги це специфікація того що повинно бути реалізовано. В них описана поведінка системи, властивості системи або її атрибути. Вони можуть слугувати обмеженнями в процесі розробки системи [1]. Такий опис є більш точним та дає

зрозуміти що за допомогою правильно сформульованих вимог та специфікацій по ним можна повністю досягнути суті, вигляд та поведінку системи що розроблятиметься, розробляється або буде розроблятися, оскільки повноцінна документація по проєкту грає роль як відправної точки так і певного стандарту до якого звертаються на всіх етапах розробки продукту. Вимоги охоплюють не лише бачення користувача та зовнішню поведінку системи, а й уявлення розробника про внутрішні характеристики. Вони описують поведінку системи у певних умовах, та характеризують властивості, які роблять систему корисною та задовольняють кінцевих користувачів.

Вимоги у своєму необробленому вигляді можуть мати форму тексту, аудіозапису або навіть відеозапису, із побажаннями замовника щодо того яким саме він бачить вирішення своєї проблеми або просто абстрактним окресленням застосунку який потенційно принесе користь. Така інформація у початковому вигляді зазвичай є хаотичним набором в більшості неоднозначних, а інколи і суперечливих тверджень та побажань, наприклад, *«Я хочу, щоб програма була інноваційною і відповідала сучасним тенденціям, а також щоб була максимально стабільною і простою в користуванні»*. Робота аналітика полягає в тому щоб зрозуміти побажання та побоювання замовника, зібрати, проаналізувати та провести декомпозицію початкових вимог, уточнити неоднозначності та усунути суперечливість у вимогах, після цього, в залежності від підходу до розробки та процесів роботи проєктної команди, деталізувати вимоги у специфікаціях структурованих таким чином щоб привнести максимальну користь проєктній команді. Від професіоналізму та рішень аналітика залежить те як менеджер проєкту оцінить час та необхідні ресурси для розробки, чи правильно розробники зрозуміють поставлене технічне завдання, чи правильно дизайнер зможе візуалізувати інтерфейс системи, яким чином тестувальники перевірятимуть якість та правильність роботи продукту, і, звісно, чи буде замовник вдоволений кінцевим результатом. Тому можна сміло стверджувати що вимоги до програмного продукту

грають роль орієнтира проєктної команди та є основною ланкою для співпраці із замовником.

Залежно від способів зберігання інформації, вимоги можуть зберігатись на фізичних та цифрових носіях, проте в сучасних реаліях ринку ІТ продуктів, враховуючи к-сть та функціональність продуктів для ведення документації, управління вимогами та управління версіями, фізичне зберігання вимог стає неактуальним та клопітним завданням. Хоча вимоги прийнято називати документацією, це необов'язково є паперова документація у класичному розумінні. В наш час це є скоріше загальною характеристикою контейнера що містить в собі специфікацію вимог, тому коли ми кажемо документ про концепцію та границі мається на увазі певне місце зберігання інформації про концепцію та границі проєкту.

Вимоги до програмного продукту поділяються за рівнями, оскільки можуть мати певний рівень деталізації та нести в собі різну цінність для певних зацікавлених сторін. Карл Вігерс та Джой Бітті пропонують поділяти вимоги на три рівні [2, с. 6]:

1. Бізнес-вимоги
2. Користувацькі вимоги
3. Функціональні вимоги

Бізнес-вимоги – це високорівнева бізнес-ціль організації чи замовників системи [2, с. 6]. Це визначення дає повністю зрозуміти суть бізнес вимог, дослівно вони є вимогами бізнесу, тобто та конкретна мета якої хоче досягнути замовник вклавшись у розробку програмного продукту. Саме ці вимоги є першочерговим питанням для обговорення із замовником.

Користувацькі вимоги – це завдання, яке певні класи користувачів повинні мати можливість виконувати в системі, або необхідний атрибут продукту [2, с. 6]. За визначенням користувацькі вимоги є більш конкретним описом можливостей системи. Деталізація може варіюватись від більш абстрактної до опису певних дій що можуть виконувати користувачі.

Функціональні вимоги – це опис необхідної поведінки системи у певних умовах [2, с. 6]. Такі вимоги є найнижчим рівнем декомпозиції та, зазвичай, включають безпосередні описи користувацьких інтерфейсів, послідовність подій що відбуваються в системі та навіть детальні описи роботи технічної інфраструктури програми.

Додатково кожна система має набір нефункціональних вимог що можуть відображати якість роботи системи, проте до цього поняття ми повернемося пізніше.

Окрім рівнів пропонується певна типізація вимог задля структуризації документації та однозначності розуміння специфікацій. На рис. 1.1 проілюстровано рівні та типи вимог, включаючи три описані вище рівні та документи в яких такі вимоги зазвичай описуються.

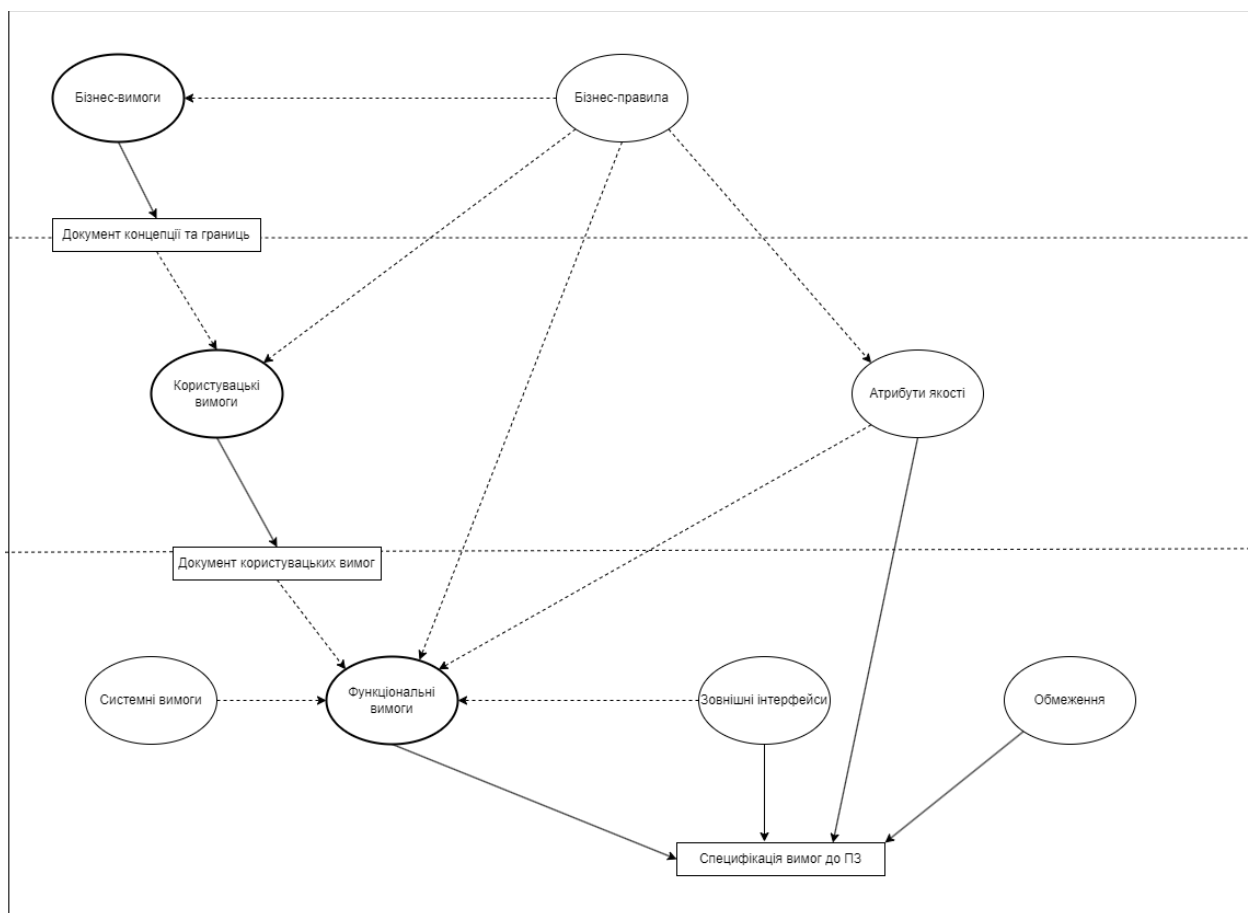


Рисунок 1.1 - Взаємозв'язки кількох типів інформації для вимог

Джерело: розроблено автором на основі [2, с. 7]

Овали позначають типи інформації вимог, а прямокутники документи, у яких зберігається ця інформація. Суцільні лінії вказують, що у зазначеному документі зберігається інформація певного типу. Пунктирна лінія вказує, що інформація одного типу є джерелом чи впливає на інформацію іншого типу або на вимогу [2, с. 7].

До першого рівня вимог - бізнес-вимог, включають бізнес-цілі та ідеї замовника, а також внутрішні та зовнішні обмеження що називають бізнес-правилами.

Бізнес-правила – це обмеження, регуляторні державні постанови, внутрішньоорганізаційні правила та стандарти що напряду впливають на можливості системи та регулюють її роботу. Такі правила варто вважати незалежними від інших вимог до продукту оскільки замовник не визначає їх, навпаки вони певним чином впливають на побажання замовника, вони створюються та змінюються поза рамок одного проєкту або продукту і досліджуються аналітиком окремо, задля дотримання відповідності чинному законодавству країни, стандартам галузі та регламенту компанії.

Документ в якому містяться вимоги бізнес рівня та бізнес-правила прийнято називати документом *про концепцію та границі* (vision and scope document) [2, с. 8]. Також на цьому рівні до документації верхнього рівня часто відносять специфікації *варіантів використання* (business cases), *статут проєкту* (project character) або *документ ринкових вимог* (market requirements document), на це впливає специфіка програмного продукту та регламент роботи команди.

Якщо продукт є комерційною системою для зовнішнього користування *документ ринкових вимог* містить в собі вимоги щодо необхідного та інноваційного функціоналу та дослідження того як саме такий функціонал допоможе продукту конкурувати з іншими.

В ситуації коли в системі передбачаються комплексні бізнес-процеси із певною специфікою, опис *варіантів використання* надає можливість формалізувати кожний етап таких процесів та закласти необхідну варіативність

поведінки системи у вигляді опису послідовних дій залучених «акторів» і подій що відбуваються в рамках дій «акторів» та самих процесів.

Після узгодження концепції та границь продукту аналітик переходить до наступного рівня – рівень користувацьких вимог. На цьому рівні взаємодія із замовником зазвичай доповнюється роботою із певними класами користувачів які, потенційно, після завершення розробки продукту, будуть ним користуватись. Зазвичай опис користувацьких вимог є описом конкретних можливостей користувача в системі, в контексті певного процесу. Хоча рівень абстракції може варіюватись, а самі вимоги можуть вміщати у собі декілька інших менш абстрактних вимог, здебільшого користувацька вимога звучить чітко та вказує на те, що саме, а як користувач повинен мати можливість зробити.

У форматі *користувацької історії* (user stories), специфікації яку часто використовують в гнучких підходах до розробки, користувацька вимога може звучати наступним чином: «Я, як оператор складу, хочу мати можливість відсортувати товари за алфавітом». В творі «Розробка вимог до програмного забезпечення» зазначають що деякі спеціалісти використовують більш широкий термін замість користувацьких вимог - «вимоги зацікавлених осіб», для позначення того факту, що вимоги будуть надаватися різними зацікавленими особами, відмінними від прямих користувачів системи [2, с. 8].

Залежно від підходу до розробки користувацькі вимоги можуть мати різний вигляд, окрім користувацьких історій що притаманні гнучким підходам існують варіанти або сценарії використання що були описані вище та таблиці «подія-відгук». Інколи такі вимоги записуються і в довільній формі проте сама суть обробки таких вимог аналітиком полягає в створенні однозначної специфікації легкої до розуміння людиною що не є спеціалістом предметної області.

Як можна побачити на рис. 1.1 бізнес-правила впливають і на користувацькі вимоги, а документ із формалізованими користувацькими вимогами називають *документ користувацьких вимог*. Слід звернути увагу що на одному рівні деталізації із користувацькими вимогами присутні атрибути якості, на які

впливають бізнес-правила, проте, які на пряму не впливають на самі користувацькі вимоги, радше вони є частиною користувацьких вимог проте в них бракує конкретики щоб віднести їх до нижньорівневих функціональних вимог.

Останнім і найнижчим рівнем, рівнем конкретики, є рівень функціональних вимог що включає в себе специфікації для розробників, дизайнерів та тестувальників із конкретними сценаріями роботи системи. На рис. 1.1 видно що бізнес-правила мають вплив і на цьому рівні вимог, це не дивно оскільки бізнес-правила можуть регламентувати і технічну і візуальну складові системи. Також на рис. 1.1 зазначені окремо *системні вимоги* (system requirements), які на пряму впливають на функціональні вимоги, хоч за стандартом ISO/IEC/IEEE 2011 пропонується достатньо вичерпне визначення таких вимог, що звучить наступним чином – системні вимоги описують вимоги до продукту, в якому міститься багато компонентів або підсистем [2, с. 9], все ж хочеться доповнити що системні вимоги також є вимогами до надійності, безпеки, продуктивності, масштабованості, інтегрованості та доступності, а також до документації та підтримки.

Такі вимоги іноді провокують створення нових форматів специфікацій, таких як *системні варіанти використання* (system use cases), та зазвичай потребують поглиблених пізнань у технічних аспектах системи та процесах розробки.

Атрибути якості (quality attributes), що визначаються на рівні користувацьких вимог, при їх формалізації на рівні функціональних вимог, набувають конкретики та окреслюють певні характеристики системи що виражаються у швидкодії та зручності. Наприклад визначена вимога щодо швидкості та надійності роботи системи, на рівні користувацьких вимог може звучати наступним чином – «Система повинна працювати швидко та стабільно». В той час як на рівні функціональних вимог потрібно надати конкретні виміри швидкості та надійності, наприклад – «Система повинна завантажувати сторінку за 0,5 секунд та результовати завантаження із помилкою не більше ніж 1 раз на 100

завантажень». Таким чином визначаються атрибути якості системи, закладаючи конкретні одниниці виміру певного атрибуту якості.

Зовнішні інтерфейси (external interfaces) виокремлені на рис. 1.1 впливають на функціональні вимоги оскільки містять прототипи дизайну елементів системи що напряду відображають те як система виглядатиме коли користувач буде працювати в ній, вони є специфікацією що надає можливість уявити працюючий продукт візуально тому при їх створенні можуть коригуватись функціональні вимоги заради розробки зручної в користуванні системи. Якщо мова йде про зовнішні інтерфейси зазвичай кажуть саме про візуальне представлення системи проте в залежності від специфіки проєкту мова може вестись про комунікаційні інтерфейси (NFC зчитувач, USB порти і тд.), інтеграція з іншими програмними системами, апаратними пристроями та користувачами.

Обмеження (constraints) є останнім елементом функціональних вимог, вони не впливають напряду на самі функціональні вимоги, а скоріше є супутніми частинами що накладають границі на можливості вибору розробників та регулюють логіку роботи системи. Прикладом використання обмеження може бути модель прав доступу користувача в системі та відповідні дозволи та обмеження що накладаються при використанні певної ролі, тобто якщо користувач із правами доступу «гість» намагатиметься змінити дані іншого користувача, система перевірятиме права його доступу та накладе обмеження якщо той не має прав доступу «адміністратора». Примінятись обмеження можуть як на технічному рівні так і на рівні бізнес-процесів, якщо вони їх потребують.

На рівні функціональних вимог самі вимоги, зовнішні інтерфейси, атрибути якості та обмеження формалізуються та записуються у *специфікації вимог до ПЗ* (software requirements specification, SRS), деталізовані на необхідному рівні в залежності від потреб проєктної команди. За стандартом ISO/IEC/IEEE 29148:2018 термін специфікації вимог до ПЗ звучить наступним чином – це структурований набір основних вимог (функцій, продуктивності, проєктних обмежень та атрибутів) програмного забезпечення та його зовнішніх інтерфейсів [3]. Описана специфікація

необов'язково повинна бути окремим документом, вимоги користувацького та функціонального рівня можна поєднувати в одному документі для створення пересічень і повноцінної інфраструктури вимог де під конкретні користувацькі вимоги будуть підв'язані відповідна їх реалізація у вигляді функціональних вимог та дизайну.

Зацікавленими сторонами (stakeholder) прийнято називати людину, групу або організацію, що активно задіяна в проєкті, на яку впливає процес або результат або яка може впливати на процес або результат [2, с. 29]. Запропоноване визначення вірно описує сторону, що залучена в проєкті та має певний вплив на нього. Зацікавлені сторони бувають зовнішніми та внутрішніми по відношенню до організації-розробника або до проєктної команди. На проєкті такими сторонами зазвичай є менеджер проєкту, аналітик, старші колеги від кожної команди розробки, замовник та користувачі що приймають участь у виявленні вимог. На рівні організації зацікавленими сторонами можуть бути: відділ маркетингу, топ-менеджери організації, інша компанія що інвестує в проєкт або є його замовником і інколи навіть держава, в тому випадку коли саме держава є замовником проєкту.

Варто приділити увагу тому як саме робота аналітика із зацікавленими сторонами відображається на вимогах до програмного продукту та яку роль вони грають в процесі розробки та управління вимогами. Насправді все просто, зацікавлені сторони є основним джерелом вимог до програмного продукту, вони найбільше зацікавлені в тому щоб продукт був розроблений відповідно до побажань клієнта та користувачів. Першочерговим джерелом вимог звісно є замовник або його представник, проте в процесі створення та управління вимогами запити на зміну деяких вимог, на переоцінку пріоритетності та на додавання нового функціоналу можуть приходити від членів команди розробки та менеджерів. Будь-які зміни завжди узгоджуються із замовником, проте сама суть в тому що ініціалізувати зміни в вимогах може не тільки він.

Оскільки джерелом вимог та їх змін є зацікавлені сторони регулярна взаємодія з ними є запорукою успішної розробки, саме, потрібного функціоналу, а

часті зустрічі із замовником та користувачами допомагають забезпечити мінімальну розбіжність в очікуваннях та побажаннях із функціоналом що реалізується. Ці зустрічі можуть приймати форму інтерв'ю, співбесід, перегляду вимог, перегляду дизайну користувацького інтерфейсу, оцінки прототипу, а у разі гнучкої розробки - відгуків клієнтів на невеликі розширення функціональності системи.

Із наведених вище описів можна підсумувати наступне – значенням вимог у розробці програмного продукту є інформативна та орієнтирна складові в процесі розробки, це є формалізований запис концепції продукту та його характеристик для використання як відправної точки на початку та посилення в процесі розробки продукту. Вимоги є сполучною ланкою між проєктною командою та замовником, а також іншими зацікавленими сторонами. Вимоги грають роль формального документу із специфікаціями щодо того як, хто, коли, чому і навіщо буде працювати у продукті що розробляється. Детальна декомпозиція вимог на трьох рівнях (бізнес, користувацький та функціональний) забезпечує проєктну команду та зацікавлені сторони вичерпною документацією що охоплює усі аспекти роботи продукту, від візуального до технічного. Тому можна зазначити що необхідність у вимогах на проєктах розробки програмних продуктів є надважливою та критичною для вдалого результату. Документація вимог набуває іще більшої цінності якщо на проєкті працює професійний аналітик, котрий знайомий із прийомами роботи із вимогами, а також має відповідні комунікаційні та особистісні навички задля взаємодії із зацікавленими сторонами.

1.2 Методи розробки та управління вимогами

Методи та підходи до роботи з вимогами завжди варіюються в залежності від ідеї замовника, бізнес-цілей проєкту, підходу до розробки продукту та специфіки предметної області, проте загальні методи не сильно відрізняються від проєкту до проєкту. Загалом процес роботи з вимогами об'єднує в собі розробку та управління, та слід розглядати ці дві області окремо оскільки перша відноситься до початкових стадій проєкту, хоча може зачіпатись на будь-якій стадії, а друга включає роботу з вже готовими специфікаціями до вимог. Карл Вігерс та Джой Бітті визначають саме ці дві області та пишуть наступне – «Незалежно від методології проєкту — водоспадна, поетапна, ітеративна, гнучка чи змішана, є дії, які треба виконати щодо всіх вимог. Залежно від методології ці операції можуть виконуватися у різний час життєвого циклу проєкту та з різним рівнем глибини та деталізації.» [2, с. 16]. Це означає що розробка та управління вимогами є невід'ємними етапами роботи на проєкті, незалежно від підходів та методів розробки продукту робота над вимогами буде обов'язковою ланкою для просування.

Автори пропонують представлення структури предмету розробки технічних умов як візуалізовано на рис. 1.2.

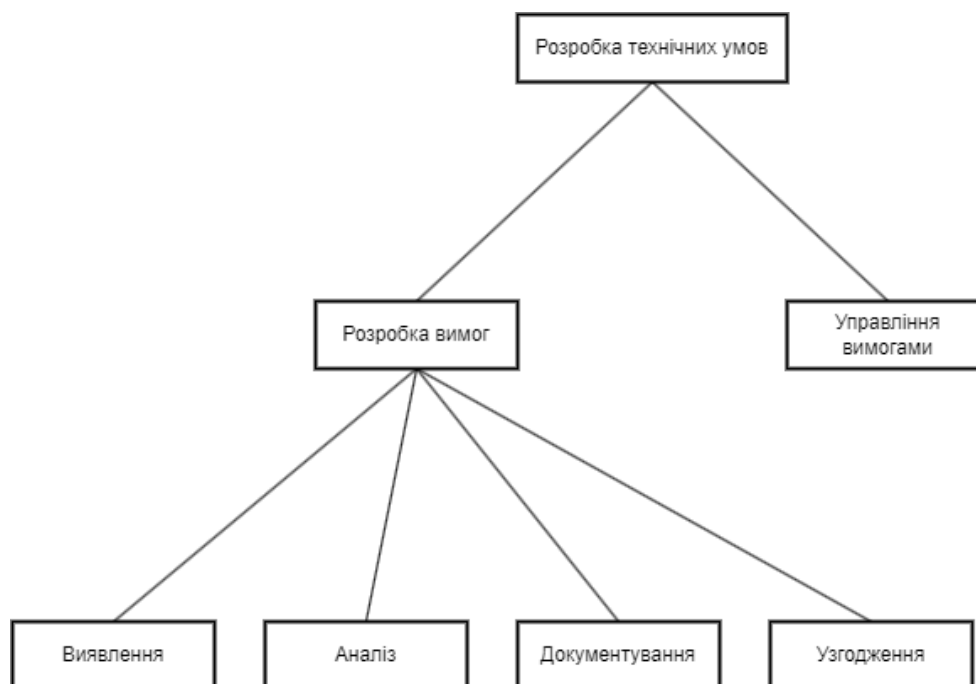


Рисунок 1.2 – Складові частини предмета розробки технічних умов

Джерело: розроблено автором на основі [2, с. 16]

Під терміном *розробка технічних умов* мається на увазі сукупність областей розробки та управління вимогами. Запропоновані на рис. 1.2 складові частини є стійким представленням етапів роботи аналітика вимог що використовують більшість експертів в сучасному світі. З права на ліво наявні такі складові частини:

1. *Виявлення* (elicitation)
2. *Аналіз* (analysis)
3. *Документування* (specification)
4. *Узгодження* (validation)

Варто розглянути так звані рекомендовані прийоми (best practices) що можуть бути використані на всіх етапах роботи з вимогами. Хоча англomовний варіант даного терміну дослівно перекладається як «кращі прийоми» стверджувати що усі перераховані нижче прийоми щодо кожної складової є кращими буде невірно. Оцінка усіх цих прийомів матиме суб'єктивний характер через різноманіття культур та регламентів роботи, підходів до планування та розробки на проєктах, а також через мінливість та непередбачуваність бажань зацікавлених сторін. Тому варто сприймати ці прийоми саме як перелік рекомендованих практик

що довели свою ефективність на тому чи іншому проєкті і застосовуються експертами у сучасній індустрії.

Першою складовою частиною є етап **виявлення вимог** що може включати такі прийоми як – інтерв'ю, співбесіди, аналіз документів, семінари, мозковий штурм, експертне опитування, створення та презентація прототипів та інші. Рекомендованими прийомами є [2, с. 49]:

- Визначення концепції продукту та границь проєкту
- Визначення класів користувачів
- Виокремлення з користувачів затятих прибічників продукту
- Створення фокус-груп
- Визначення користувацьких вимог
- Визначення системних подій та реакцій на них
- Проведення інтерв'ю для виявлення вимог
- Проведення семінарів для виявлення вимог
- Спостереження за користувачами на робочих місцях
- Роздача опитувальних листів
- Аналіз документів
- Вивчення звітів про проблеми
- Повторне задіяння існуючих вимог

Явно видно що ціль роботи аналітика на етапі виявлення вимог полягає у ідентифікації джерел вимог, безпосередньому зборі вимог, визначенні потенційних користувачів та прихильників продукту, оцінці середовища в якій буде розроблятися продукт та аналізі вже існуючої документації та зібраної інформації. На цьому етапі аналітик записує вимоги у довільній формі щоб зафіксувати побажання клієнта, на наступних етапах таку необроблену інформацію він перетворює в формалізовані знання. Слід пам'ятати що не кожна із практик щодо виявлення вимог може знадобитись на конкретному проєкті, деякі із компаній можуть не мати ресурсів для залучення окремих користувачів задля зняття вимог, проходження опитувань

або тестування продукту. В інших випадках стається так що в організації не регламентовано організація окремих семінарів для виявлення вимог, також коли аналітик працює над вимогами до нового проєкту останні три пункти в переліку відкидаються через відсутність наявної документації та даних.

Серед перелічених практик що використовуються при виявленні вимог варто детальніше розглянути такі:

1. Визначення класів користувачів
2. Створення фокус-груп
3. Проведення семінарів для виявлення вимог

Суть та використання даних практики не одразу стає явною, тому варто відповісти на питання: які цілі їх використання, як саме їх використовувати, та яку користь вони несуть.

Визначення класів користувачів є практикою націленою на виокремлення, класифікацію та групування потенційних або реальних користувачів системи, задля виявлення вимог у представників кожного класу користувачів. Аналітик опитує клієнта щодо того хто на його думку буде використовувати систему, на основі цього аналітик придумує стільки класів користувачів скільки може для охоплення всіх можливих потреб користувачів системи. Далі класи групуються за схожими чи ідентичними потребами, а нерелевантні класи викреслюються. На основі отриманого списку класів, аналітик, клієнт та інші зацікавлені особи можуть покрити вимогами усі потенційні потреби кожного класу користувачів, а за наявності представників кожного класу аналітик має можливість напряду зняти вимоги щодо функціоналу який нестиме користь для конкретних користувачів системи.

Створення фокус-груп типових користувачів націлене на виявлення подробиць щодо функціональності та якісних характеристик схожих продуктів або попередніх версій поточного продукту. Хоча в компанії не завжди є ресурси для організації таких зустрічей, вони можуть бути особливо цінними при розробці комерційних та масмаркет продуктів. Зазвичай навідміну від прихильників

продукту фокус-групи не мають повноважень для прийняття рішень. Аналітик організовує заходи де зустрічається із членами фокус-групи попередньо підібраними для обговорення певної теми або функціоналу. На таких заходах рекомендується інтерактивна взаємодія із користувачами для налагодження зворотнього зв'язку та детального обговорення теми зустрічі. Зустрічі із фокус-групами надають аналітику можливість виявити вимоги конкретних класів користувачів що могли бути упущені або приховані під час їх збору у клієнта.

Проведення семінарів для виявлення вимог є ще однією практикою що включає організацію зустрічей проте в даному випадку із зацікавленими сторонами, заради сумісної роботи таких осіб над виявленням вимог. Елен Готтендінер (Ellen Gottesdiener, 2002) визначає семінар з виявлення вимог як «структуровану нараду, на якій спеціально відібрана група зацікавлених осіб та експертів працює спільно над визначенням, створенням, уточненням та досягненням згоди щодо очікуваних артефактів (таких, як моделі та документи), що представляють користувацькі вимоги» [4]. Це твердження детально описує процес роботи учасників семінару під час зустрічі, проте дає хибне представлення щодо того вимоги якого рівня можуть обговорюватись. В залежності від складу групи, теми семінару та специфіки функціоналу, учасники семінару можуть виявляти вимоги будь-якого рівня, а також бізнес-правила, атрибути якості, та обмеження і дизайн. Даний прийом варто використовувати досвідченим членам проєктної команди оскільки він вимагає не тільки гарних організаційних навичок, але й комунікаційних, асертивних та навичок публічного виступу. Як результат проведення семінарів аналітик отримує список більш детальних та вичерпних вимог різних рівнів, а також, в залежності від ходу зустрічі, вирішує суперечливості в функціоналі та розумінні проєктної команди.

Другою складовою є **аналіз вимог** що є етапом отримання більш широкого і точного розуміння усіх вимог і представлення їх у різних формах. Набір рекомендованих практик виглядає наступним чином [2, с. 49]:

- Моделювання середовища програми

- Створення прототипів
- Аналіз здійсненності
- Розстановка пріоритетів вимог
- Створення словника даних
- Моделювання вимог
- Аналіз інтерфейсів
- Розподіл вимог за підсистемами

Як видно зі списку діяльність аналітика на етапі аналізу вимог полягає у глибокому дослідженні зібраних вимог, бачення та побажань клієнта та зацікавлених сторін. До наданого переліку також можна додати такі пункти:

- Розкладання високорівневих вимог до необхідного рівня деталізації
- Виявлення прогалин у вимогах або зайвих вимог, що не відповідають заданим границям проекту

Ці дві практики включені до набору інструментів що використовує аналітик під час дослідження вимог, проте варто приділити увагу цим прийомам оскільки саме вони є основою формування вичерпних, структурованих вимог.

Етап аналізу вимог дає можливість деталізувати усі виявлені вимоги, усунути протиріччя та неоднозначності в них, за допомогою емпіричних методів дослідження виявити приховані вимоги, можливі обмеження та приблизні атрибути якості, а також надає можливість спланувати пріоритети функціоналу для розробки. Цей етап є найбільш відповідальним та творчим у роботі аналітика оскільки саме на ньому спеціаліст не тільки використовує здобуті навички та досвід але й підходить до оцінки неформально та з креативністю.

Серед перелічених практик що використовуються при аналізі вимог варто детальніше розповісти про такі:

1. Моделювання середовища програми
2. Створення прототипів
3. Розстановка пріоритетів вимог

Суть та використання перших двох практики не одразу стає явною, а детальний огляд третьої просто необхідний оскільки розкриває повноту її впливу на проєктний менеджмент, тому варто відповісти на питання: які цілі їх використання, як саме їх використовувати, та яку користь вони несуть.

Моделювання середовища програми включає в себе прийоми створення контекстних діаграм, мапи екосистеми, діаграм сутність-зв'язок та інших графічних представлень середовища та системи що розроблятиметься. Змодельована контекстна діаграма середовища та мапа екосистеми включається до специфікації документу концепції та границь для спрощеного представлення усього продукту в графічному вигляді, це надає можливість швидше зрозуміти та запам'ятати особливості архітектури, зв'язків всередині системи та та зв'язків підсистем, а також функціоналу продукту, як новим членам проєктної команди так і колегам яким потрібно освіжити знання.

Створення технічних прототипів та користувацьких інтерфейсів надає можливість графічно або контекстно візуалізувати вимоги. Якщо клієнт, користувач або зацікавлена сторона не впевнені в певній вимозі, аналітик може створити прототип інтерфейсу або технічного ланцюжка, та презентувати його відповідній зацікавленій особі, якщо бачення особи співпадають із представленим баченням аналітика це є приводом для підтвердження узгодження відповідної вимоги. Проте якщо уявлення зацікавленої сторони є відмінним від запропонованого прототипу, це є приводом для уточнення старих та виявлення нових вимог щодо цього функціоналу. Таким чином після обробки уточнень та нових вимог аналітик матиме прототип продукту на котрий можна буде посылатись як на повноцінний прообраз системи.

Розстановка пріоритетів вимог зачіпає не тільки область відповідальності аналітика але й проєктного менеджера, оскільки в обов'язки другого входить організація та планування роботи проєктної команди протягом усього циклу розробки. Зазначена практика включає співпрацю аналітика із проєктним менеджером для аналізу та визначення відносних пріоритетів функціоналу що

розроблятиметься, на основі визначених пріоритетів встановлюється те на якій ітерації або в якій версії продукту той чи інший функціонал або вимога буде реалізована. На проєктах гнучкої розробки пріоритети можуть часто коригуватися у відповідності до потреб клієнта, зовнішніх та внутрішніх умов, а також бізнес-цілей. В результаті визначення пріоритетів та планування випуску продукту проєктна команда матиме чітке уявлення про те що саме буде розроблятися в першу чергу, а що буде відкладено на пізніші ітерації.

Третьою складовою частиною представлено **документування вимог** що являє собою формалізацію та зберігання вимог структурованим та організованим способом для зручного доступу до них. Набір рекомендованих прийомів на які варто звернути увагу [2, с. 49]:

- Використання шаблонів специфікацій вимог
- Визначення джерел вимог
- Призначення кожній вимозі унікального ідентифікатора
- Документування бізнес-правил
- Визначення нефункціональних вимог

Метою цього етапу роботи є перетворення зібраних вимог у письмові специфікації та діаграми, придатні для розуміння, аналізу та використання як проєктною командою так і зацікавленими сторонами. Використання шаблонів та ідентифікація вимог є вкрай ефективними прийомами що прискорюють роботу аналітика та ознайомлення проєктної команди із вимогами, а також допомагають організувати цілу інфраструктуру для документації.

Документування вимог може вестись як у паперовому, аудіо та відео форматах так і в цифрових текстових ресурсах та в графічному форматі. Даний етап роботи аналітика хоч і є більш формальним, проте має точки розширення та надає спеціалісту можливість проявити свої організаційні навички та навички структуризації документації.

Слід виокремити та розглянути такі практики:

1. Визначення джерел вимог

2. Призначення кожній вимозі унікального ідентифікатора
3. Визначення нефункціональних вимог

Визначення джерел вимог, на перший погляд, може здатись незначною деталлю та формальністю якою не варто нехтувати. В процесі документування вимог будь-яких рівнів та супутніх атрибутів продукту аналітику необхідно зазначати джерело конкретної вимоги задля впевненості в тому що всі зацікавлені особи розуміють навіщо необхідне та або інша вимога. Запис джерел вимог допомагає відслідковувати звідки ця вимога поступила і в разі надходження змін що зачіпають цю вимогу, знайти зацікавлену особу до якої слід звернутись. Документування джерел є корисним інструментом організації та структуризації інформації в специфікаціях вимог.

Призначення кожній вимозі унікального ідентифікатора також як і визначення джерел вимог є ефективним інструментом організації інформації, оскільки допомагає ідентифікувати та відслідковувати вимоги. Якщо в організації немає погодженого протоколу ідентифікації вимог аналітик повинен розробити його та запропонувати для впровадження. Узгоджений інструментарій повинен забезпечувати можливості доповнення, зміни та видалення вимог. Також при використанні ідентифікаторів вимог завдання відслідковування статусу конкретної вимоги спрощується в кілька разів. Як результат вимогами із унікальними ідентифікаторами стає простіше керувати, а також стає простіше відслідковувати зміни в конкретних вимогах.

Визначення нефункціональних вимог або атрибутів якості продукту є завданням збору та аналізу вимог що описують загальні характеристики роботи системи, такі як – швидкодія, надійність, стабільність, зручність у використанні, візуальна привабливість та інші. Аналітику при використанні даного прийому необхідно відійти від питань функціональності та поринути у можливі якісні характеристики системи що базуються на побажаннях та баченні клієнта, навіть якщо такі побажання мають виключно суб'єктивний характер.

Останньою та заключною складовою є **узгодження вимог**, цей етап забезпечує перевірку відповідності вимог до поставлених бізнес-цілей та уявлень клієнта. Рекомендовані прийоми при узгодженні вимог [2, с. 49]:

- Дослідження документів з вимогами
- Тестування вимог
- Визначення критеріїв приймання
- Моделювання вимог

В процесі розробки продукту перевірку вимог може проводити як і сам автор, тобто аналітик, так і члени проєктної команди та навіть замовник, залежно від регламенту процесів розробки. Оскільки при плануванні роботи над вимогами передбачається не один цикл перевірки вимог, аналітик повинен бути готовим до монотонної повторюваної роботи. Задля уточнення деталей вимог більш верхнього рівня та підтвердження правильності вже написаних вимог, процес узгодження може повторюватись n -ну кількість разів, але витрати часу на таку роботу не повинні виходити за розумні границі, інакше робота проєктної команди може сповільнюватись, а інколи навіть блокуватись.

Моделювання вимог є прийомом із наведеного вище списку що варто виділити, оскільки він не так часто застосовується в роботі аналітика проте привносить значну користь при роботі з вимогами. За допомогою окремого програмного забезпечення створеного безпосередньо для моделювання прототипів системи аналітик може створювати макети системи на основі вже зібраних вимог, задля того щоб користувачі могли побачити та обрати дизайн, перевірити можливу реалізацію функціоналу, та, звісно, відстежити виконання вимог. Аналітику що планує використовувати такі програмні продукти необхідно мати відповідні навички та досвід, оскільки кожний етап розробки вимог займає час що є одним із найважливіших ресурсів. Навчання роботі в таких програмах в процесі розробки вимог є недопустимою розкішшю.

Управління вимогами на відміну від розробки вимог вимагає вже наявної документації в яку, в процесі розробки, вносяться зміни. Основною метою

управління вимогами є адаптація до можливих реальних змін задля зниження їх руйнівного ефекту на проєкт. До рекомендованих прийомів по управлінню вимогами відносять [2, с. 50]:

- Визначення процесу управління змінами
- Аналіз впливу змін
- Визначення базової версії наборів вимог та управління версіями вимог
- Відстеження хронології змін
- Відстеження стану вимог
- Відстеження проблем з вимогами
- Створення матриці зв'язків вимог
- Використання засобу управління вимогами

Як видно з викладеного вище, управління вимогами є процесом націленим на роботу із готовими вимогами, що вже пройшли узгодження або, навіть, були реалізовані, або із частиною вимог, для того щоб оптимізувати роботу зі змінами які неминуче будуть вноситись. Аналітик при використанні прийомів управління вимогами знаходить приховані зв'язки які раніше були пропущені, а на етапі аналізу впливу змін на проєкт може навіть дійти висновку що певний функціонал стане неактуальним в контексті доданих змін. Тому варто наголосити що управління вимогами включає не лише узгодження та фіксацію змін але й аналіз впливу змін на продукт та проєкт, а також оновлення плану проєкту враховуючи ці зміни. Регламентований процес управління змінами допомагає аналітику та команді спростити процеси аналізу та внесення змін, налагодити зв'язок щодо змін між зацікавленими сторонами, а також визначити чіткий протокол дій для уникнення хаосу в процесі роботи, що може призвести до блокування роботи проєктної команди.

В рекомендованих практиках що використовуються при управлінні вимогами присутні декілька прийомів які необхідно розгорнуто описати заради створення чіткішого уявлення про їх суть, використання та кінцевий результат.

Такими практиками є:

1. Визначення базової версії наборів вимог та управління версіями вимог
2. Відстеження проблем з вимогами
3. Створення матриці зв'язків вимог

Визначення базової версії наборів вимог та управління версіями вимог є прийомом що включає в себе фіксацію версії вимог для певного відрізка розробки та використання попередньо узгодженого процесу управління вимогами. Тобто для певної ітерації розробки аналітик повинен узгодити відповідний набір готових вимог що буде використаний як базова версія, а в процесі розробки, за необхідності, притримуючись регламенту процесу управління вимогами, вноситиме в них зміни. Якщо під час розробки у клієнта виникатимуть питання чи правильно було реалізовано або описано той чи інший функціонал, повернення до базової версії вимог забезпечить ясність у розумінні. В сучасних програмних продуктах для управління вимогами та версіями є вбудовані інструменти відслідковування версій, тому робота аналітика спрощується в цьому аспекті.

Відстеження проблем з вимогами може бути достатньо складним завданням особливо в великих компаніях чи на складних проєктах. Для використання даної практики необхідно залучати окремий людський ресурс що може бути недоступний, також важливо зазначити, що пояснити раціональність витрат часу та сил на відстеження проблем не завжди вдається, що виливається в те що аналітик повинен сам відслідковувати і виправляти усі проблеми та закривати прогалини по вимогах. Існують засоби відслідковування дефектів проте не завжди організація має в розпорядженні такі сервіси. Якщо аналітик буде використовувати цей прийом при управлінні вимогами, до того ж якщо матиме в розпорядженні осіб яких може призначити відповідальними за відслідковування відповідних проблем, то як результат стан описаних вимог поліпшиться, а команда буде впевнена в тому що посилається на перевірену документацію в процесі розробки.

Створення матриці зв'язків вимог це практика що включає документування зв'язків усіх вимог та супутніх артефактів для відстеження

можливих впливів при внесенні змін та для перевірки реалізації описаного функціоналу. Оскільки при створенні матриці зв'язків аналітик записує усі наявні вимоги та зв'язки між ними з'являється можливість відстежити залежності на усіх рівнях вимог та залежності дочірніх вимог із вимогами верхніх рівнів що їх породжують. Аналітик вимог на проєкті зобов'язаний вести таку таблицю в процесі розробки вимог задля своєчасного її оновлення та підтримання в актуальному стані.

1.3 Основні принципи системного підходу до управління вимогами

Системний підхід - один із головних напрямів методології спеціального наукового пізнання та соціальної практики, мета і завдання якого полягає у дослідженнях певних об'єктів як складних систем. Системний підхід сприяє формуванню відповідного адекватного формулювання суті досліджуваних проблем у конкретних науках і вибору ефективних шляхів їх вирішення [5]. Таке визначення системного підходу є більш загальним та не стосується конкретно спеціалізованої низки прийомів щодо роботи з вимогами до програмного продукту, проте несе в собі суть підходу до організації процесу управління вимогами. Іншими словами системний підхід дозволяє розглянути такий процес як повноцінну складну систему, провівши декомпозицію цієї системи можна відслідкувати залежності між її елементами та проаналізувати їх стан, а також виявити внутрішні та зовнішні фактори впливу. Після цього необхідно зробити відповідні висновки щодо стану системи загалом та запропонувати варіанти для оптимізації її ефективності.

В сучасних джерелах достатньо вичерпно описані принципи системного підходу, проте, оскільки цей підхід описаний як загальний метод дослідження, ці принципи описані поза контексту управління вимогами, через що деякі із них не застосовуються або модифікуються у відповідності до специфіки предметної області.

Пропонуються такі принципи системного підходу як основні, в контексті управління вимогами [5, 6, 7]:

1. **Цілісність:** розгляд системи як єдиного цілого, де кожен елемент має своє місце і роль
2. **Взаємозв'язок елементів:** вивчення взаємодії між компонентами системи
3. **Ієрархічність:** аналіз системи на різних рівнях складності
4. **Невизначеність:** дослідження системи із врахуванням та адаптацією під непередбачувані події або зовнішні впливи
5. **Динамічність:** урахування в системі часових характеристик змін

Необхідно зазначити що системний підхід в управлінні вимогами, зазвичай, не є явним для тих хто його використовує, оскільки в першу чергу спеціалісти звертають увагу на логіку роботи із вимогами лише в контексті безпосереднього застосування відповідних прийомів, і тому для них не є очевидним те що, використавши звичайне здорове мислення вони використовують саме системний підхід.

Розглянемо основні принципи системного підходу в контексті використання їх для управління вимогами.

Перший принцип системного підходу – **цілісність**, полягає у дослідженні усіх зібраних вимог як частини єдиного цілого. Це означає, що кожна вимога має бути проаналізована в контексті всієї системи, а не окремо. Такий підхід забезпечує повне розуміння взаємозв'язків і залежностей між вимогами, що дозволяє уникнути конфліктів між ними та забезпечити їх узгодженість. Завдяки такому підходу, аналітик вимог може створити вичерпний та збалансований набір вимог до програмного продукту, де кожна з них підтримує і доповнює інші, сприяючи

досягненню загальної мети проєкту. Це особливо важливо у великих та складних системах, де конфлікти між вимогами можуть призвести до значних затримок в розробці та перевитрат ресурсів. Тому, цілісність сприяє не лише ефективному управлінню вимогами, але й підвищує загальну якість та успішність проєкту. Як приклад застосування даного принципу можна навести ситуацію при розробці складної інформаційної торговельної системи. У цьому випадку вимоги до функціональних модулів, таких як торгові процедури, складський облік, відстеження постачання та інші, повинні розглядатися як частина єдиного цілого. Якщо вимоги до кожного модуля розроблятимуться окремо без врахування їх взаємодії, можуть виникнути конфлікти, такі як несумісність даних або дублювання функцій. Наприклад якщо модуль функціоналу із профілем користувача матиме одну модель даних адреси, а в модулі складського обліку буде використана інша модель, це призведе до помилок в проведенні торгових процедур через невідповідність формату даних. Застосування принципу цілісності дозволяє врахувати взаємозв'язки між цими модулями, забезпечити єдині стандарти та уникнути подібних проблем, що сприятиме ефективній роботі користувачів в системі, а також відсутності колізій між розумінням розробників в процесі кодування.

Взаємозв'язок елементів є наступним принципом системного підходу, що полягає в аналізі взаємодії між окремими вимогами та іншими компонентами системи. Це означає, що під час розробки та управління вимогами необхідно враховувати, як кожна вимога впливатиме на інші компоненти системи і навпаки. Такий підхід допомагає виявити потенційні конфлікти та синергію між вимогами, що дозволяє більш ефективно розробляти документацію. Взаємозв'язок елементів забезпечує глибоке розуміння того, як зміни в одній вимозі можуть вплинути на інші вимоги та компоненти системи, що є критично важливим для написання структурованої документації та створення працюючої системи. Практичним прикладом застосування принципу взаємозв'язку елементів може бути впровадження багатофакторної аутентифікації в додатку для задоволення потреби

в підвищенні безпеки. Таку вимогу варто розглянути у контексті її впливу на користувацький досвід роботи в системі. Якщо додати багатofакторну аутентифікацію, це може ускладнити процес входу в систему для користувачів, що, в свою чергу, може знизити задоволеність користувачів і частоту використання додатку. Таким чином, необхідно врахувати взаємозв'язок між вимогою до безпеки і вимогою до зручності використання. Як рішення, можна знайти оптимальний баланс між безпекою і зручністю, наприклад, впровадивши адаптивну аутентифікацію, яка застосовується тільки в разі підозрілої активності. Це забезпечить високий рівень безпеки без значного впливу на зручність користування додатком.

Принцип ієрархічності передбачає структурування вимог за рівнями пріоритетності та деталізації. Інакше кажучи це означає, що усі вимоги повинні бути класифіковані за їхньою значущістю і рівнем необхідної деталізації, що дозволяє краще організувати процес планування розробки та допомагає пояснити членам команди розробки чому певні вимоги є більш пріоритетними для реалізації. Такий підхід допомагає зосередитися на найбільш критичних вимогах на початкових етапах проєкту і поступово переходити до менш важливих функцій згодом. Ієрархічне структурування також поглиблює розуміння членів проєктної команди щодо загальної концепції та функціоналу продукту, оскільки кожен знає, які вимоги є першочерговими і як вони взаємопов'язані з іншими елементами системи. Таким чином, принцип ієрархічності сприяє більш ефективному управлінню складними вимогами і їхньою пріоритизацією, що підвищує рівень якості процесу планування та розробки. Як приклад застосування цього принципу можна уявити ситуацію коли після узгодження набору вимог до нового продукту – вільного торгового майданчика, члени проєктної команди розподіляють їх за пріоритетом реалізації. Як результат вимоги можуть бути розділені на кілька рівнів:

- основні функції (наприклад, реєстрація нового користувача в системі)

- додаткові функції (наприклад, інтеграція з іншими корпоративними додатками)
- незначні покращення (наприклад, додавання підказок в інтерфейсі)

На початковому етапі команда зосереджується на реалізації основних функцій, оскільки вони є критичними для користувачів та життєвонеобхідними для правильного функціонування системи. Після завершення роботи над основними функціями, команда переходить до впровадження додаткових функцій, що розширюють загальну функціональність програмного продукту. Нарешті, на завершальному етапі проєкту, команда реалізує незначні покращення, які роблять продукт більш привабливим і зручним у використанні. Таке застосування ієрархічного принципу дозволяє ефективно керувати складністю вимог і забезпечує послідовне досягнення цілей проєкту.

Принцип невизначеності полягає в тому, що в більшості випадків досліджується система, про яку не все відомо, поведінка якої не завжди зрозуміла, невідома її структура, непередбачуваний перебіг процесів, невідомі зовнішні впливи тощо. Це означає, що під час управління вимогами часто доводиться працювати з неповною інформацією і враховувати можливість непередбачуваних змін або подій. Такий підхід вимагає від команди гнучкості, здатності адаптуватися до нових обставин і приймати рішення на основі наявної інформації, незважаючи на її неповноту. Практичним прикладом застосування принципу невизначеності може бути управління вимогами під час розробки нового продукту на ринку, який швидко змінюється або в умовах економічної нестабільності. Наприклад, розробка мобільного додатку для фінансових послуг в умовах нестабільних економічних умов і за постійної зміни регуляторних законів. В такій ситуації, проєктна команда повинна бути готовою до того, що вимоги можуть змінитися через нові законодавчі акти, зміни на ринку або зміну пріоритетів функціоналу замовником. Це може включати нові вимоги до безпеки, зміну уподобань користувачів, появу нових конкурентів з інноваційними рішеннями або новий технологічний прорив в індустрії. Застосування принципу невизначеності допомагає команді бути готовою

до таких змін, планувати резервні варіанти і швидко адаптуватися до нових умов, що в кінцевому підсумку підвищує конкурентноспроможність продукту. Конкретно цей принцип гармонійно вписується у гнучкий підхід до розробки програмних продуктів.

Останнім є **принцип динамічності**, суть якого полягає в необхідності врахування змін у вимогах з часом та адаптивності до нових умов. Це означає, що вимоги до системи або проекту можуть змінюватися на різних етапах його життєвого циклу через нові технологічні можливості, зміни ринкових умов або інші зовнішні і внутрішні фактори. Для ефективного управління такими змінами необхідно впровадити методи управління та зміни вимог, що дозволяють оперативно реагувати на нові виклики і забезпечують актуальність і відповідність вимог поточним умовам. Динамічність допомагає уникати застою в процесі розробки і забезпечує постійну відповідність кінцевого продукту потребам користувачів та замовника. Певним чином цей принцип схожий із принципом невизначеності, проте він несе в собі трохи відмінну філософію управління вимогами, оскільки зосереджується на постійній адаптації і еволюції вимог у відповідь на відомі зміни, тоді як принцип невизначеності більше орієнтується на роботу з невідомими і непередбачуваними факторами. Слід навести практичний приклад застосування принципу динамічності, таким прикладом може бути розробка програмного забезпечення для електронної комерції. У цьому контексті вимоги можуть змінюватися через введення нових платіжних систем, зміну законодавства щодо захисту персональних та комерційних даних або нові тенденції в користувацькому досвіді. Під час розробки проєктна команда може стикнутися з необхідністю інтеграції нової платіжної системи, яка набула популярності серед користувачів зовсім недавно, або з необхідністю впровадження нових функцій для покращення зручності навігації додатком. Врахування принципу динамічності дозволяє команді швидко адаптуватися до цих змін, оновлюючи вимоги та коригуючи план розробки відповідно до нових умов.

У підсумку можна констатувати наступне - системний підхід до управління вимогами є ефективним прийомом для успішної розробки програмних продуктів, оскільки він забезпечує всебічне розуміння і врахування усіх аспектів вимог як частини єдиного цілого. Цей підхід дозволяє ефективно виявляти та вирішувати потенційні конфлікти між вимогами, забезпечувати їх узгодженість, правильно структурувати і пріоритизувати вимоги, а також адаптуватися до змін і непередбачуваних обставин. Завдяки цьому підходу процеси проєктного управління та управління вимогами стають більш гнучкими, керованими, ефективними та прозорими, оскільки він забезпечує повну відповідність кінцевого продукту потребам замовника, користувачів і ринковим умовам, навіть у швидкозмінному та нестабільному середовищі.

РОЗДІЛ 2

АНАЛІЗ ДІЯЛЬНОСТІ З РОЗРОБКИ ПРОГРАМНИХ ПРОДУКТІВ У КОМПАНІЇ ТОВ «ПЛАТМАСТЕР»

2.1 Загальна характеристика компанії

Товариство з обмеженою відповідальністю «Платмастер» є невеликою організацією, що була заснована в 2018му році в місті Київ із метою проєктування, створення та підтримки ряду програмних продуктів для зовнішнього загального користування. Директором та засновником компанії є Бугайов Микола Вікторович. Основним напрямом діяльності організації за КВЕД є 63.11 Оброблення даних, розміщення інформації на веб-вузлах і пов'язана з ними діяльність. Що на пряму відповідає фактичній діяльності організації якою є розміщення розроблених веб-додатків на веб-вузлах в мережі інтернет та оброблення вхідних даних користувачів при роботі із такими додатками організації.

Як додаткові види діяльності зазначені такі види за КВЕД [8]:

- 46.11 Діяльність посередників у торгівлі сільськогосподарською сировиною, живими тваринами, текстильною сировиною та напівфабрикатами
- 46.12 Діяльність посередників у торгівлі паливом, рудами, металами та промисловими хімічними речовинами
- 46.13 Діяльність посередників у торгівлі деревиною, будівельними матеріалами та санітарно-технічними виробами
- 46.14 Діяльність посередників у торгівлі машинами, промисловим устаткуванням, суднами та літаками

- 46.18 Діяльність посередників, що спеціалізуються в торгівлі іншими товарами
- 46.19 Діяльність посередників у торгівлі товарами широкого асортименту
- 58.29 Видання іншого програмного забезпечення
- 62.01 Комп'ютерне програмування
- 62.09 Інша діяльність у сфері інформаційних технологій і комп'ютерних систем
- 63.12 Веб-портали
- 63.99 Надання інших інформаційних послуг, н.в.і.у.
- 66.11 Управління фінансовими ринками
- 73.12 Посередництво в розміщенні реклами в засобах масової інформації
- 73.20 Дослідження кон'юнктури ринку та виявлення громадської думки

Як бачимо більшість додаткових видів діяльності компанії націлена на посередницьку торгівлю товарами різного призначення, здебільшого, сільськогосподарською, промисловою та будівельною сировиною, а також товарами широкого асортименту. З цього переліку можна зробити висновок що метою розробок компанії є переведення торгових процесів різних галузей та ринків у цифрову площину із подальшою оптимізацією часу роботи користувача та спрощенням таких процесів для прискорення проведення транзакцій між постачальниками та споживачами.

Оскільки архітектура побудови програмних продуктів що розробляються в ТОВ «Платмастер» дозволяє створювати клони окремих рішень, організація також має можливість продавати такі комплексні рішення як окремі продукти для зовнішніх замовників. Якщо звернути увагу на останні два види діяльності у списку, можна припустити що організація планує просувати продукти та рішення інших підприємств у своїх системах заради додаткового прибутку та мережевого ефекту, що може надати додатковий приріст в базі користувачів при просуванні їх власних продуктів. Щодо останнього пункту варто зазначити що в планах компанії є організація системи business intelligence у вже запусчених продуктах для надання

користувачам можливості збору необхідної значущої інформації та створення на її основі аналітичних звітів та графічних представлень різної складності та змісту, таке рішення дозволить не тільки користувачам використовувати інструменти аналітики але й самій організації збирати та аналізувати важливу інформацію щодо ринку та роботи користувачів в системі, і як результат створювати нові та оновлювати вже наявні рішення задля збільшення ефективності та зручності продукту.

Товариство «Платмастер» має горизонтальну або пласку організаційну структуру розподілу влади, тобто компанія має короткий ланцюжок командування, як зображено на рис. 2.1. Це дозволяє співробітникам зосереджуватись і витрачати час на глобальні цілі компанії, оскільки вони мають більше автономії, свободи та незалежності для виконання своїх ролей і обов'язків [9]. Оскільки робота та зусилля в невеликій компанії відносно прозорі, повноваження щодо прийняття рішень розподіляються, і, як результат, працівники несуть відповідальність за свої рішення.



Рисунок 2.1 – Схематичне зображення горизонтальної організаційної структури [9]

Організаційна структура виробничого процесу, в свою чергу, є проектною. Суть проектної структури полягає в створенні спеціальної проектної команди фахівців на період виконання проекту. В даному випадку це команда розробників та менеджер проекту відповідно. Кожен співробітник, задіяний в проекті, відповідає за одну функціональну область. Проект реалізується під керівництвом

проектного менеджера, який знаходиться в прямому підпорядкуванні генеральному директору компанії. Проектна команда не орієнтується на загальноорганізаційні цілі і націлена на виконання конкретних проектних завдань [10].

Наразі в ТОВ «Платмастер» нараховується до 50ти співробітників тому змінювати організаційну структуру не є доцільним, проте варто зазначити що за теперішньої структури на співробітників покладається додаткова відповідальність за свої рішення, тому при залученні нових кадрів важливими факторами є досвід та навички, це забезпечує впевненість у рішеннях що приймаються на робочому місці.

Оскільки компанія була заснована відносно недавно, а штат співробітників невеликий, варто сказати про те що чітких корпоративних політик та протоколів наразі не регламентовано що, звісно, є мінусом для організації процесів управління, проте є великим плюсом для самих робітників, оскільки вони мають більшу незалежність та свободу в плануванні та організації свого робочого дня. Відсутність регламентованих процесів та протоколів роботи створюють певний буфер невизначеності при плануванні робіт проектної команди, проте також додають відчуття комфорту та свободи що створюють настрій товарищескості та великої сім'ї. Важливо зазначити що відсутність строгих протоколів роботи несуть за собою ряд організаційних проблем, наприклад, відсутність співробітників на робочих місцях без попереднього узгодження, порушення процесу планування та управління проектом, а також збір вимог без присутності аналітика.

Ключовими партнерами організації є такі компанії як Helsi, Document.online, Держзакупівлі.онлайн та інші. Потенційними клієнтами є малий, середній та великий бізнес України, а в майбутньому і країн світу, що шукає інноваційних рішень із цифровізації бізнес-процесів торгівлі та постачання товарів, послуг та робіт.

Наразі товариство з обмеженою відповідальністю «Платмастер» випустило у загальний доступ для публічного бета-тестування лише один продукт під назвою «Platmix» тому користувацька база, поки що, на початковому етапі зародження.

Обсяг послуг що надаються можна буде виміряти згодом, після повноцінного випуску продукту в робочу середу. Незважаючи на те, що компанія ТОВ «Платмастер» перебуває на початковому етапі свого розвитку та це її перший продукт, вона активно працює над його впровадженням на ринку. Завдяки випуску продукту для бета-тестування, компанія встановлює контакт з потенційними клієнтами і збирає важливі відгуки щодо функціоналу та якості продукту. Це дозволяє вдосконалити продукт перед його офіційним виходом на ринок. Позиція на ринку наразі може бути визначена як початкова, але з можливістю для подальшого росту та розвитку, зокрема, за рахунок врахування відгуків користувачів та вдосконалення продукту відповідно до потреб ринку.

Оскільки компанія ще не заявила про себе на ринку та, наразі, немає повноцінних випущених продуктів, компанія ще не має значних досягнень або відзнак у своїй галузі. Проте, вона активно працює над створенням і вдосконаленням свого продукту та планує здобути визнання завдяки інноваційним підходам та високій якості своєї роботи.

Компанія ТОВ «Платмастер» хоча і не має великого штату співробітників та регламенту корпоративних політик, все ж притримується певних принципів та стратегій що закладені в основі діяльності організації.

1. **Розвиток та підтримка команди.** Компанія активно інвестує у розвиток своїх співробітників, надаючи можливості для навчання на робочому місці та отримання вищої освіти. Вона створює сприятливу атмосферу для навчання та підтримки, щоб надихати молодих та недосвідчених працівників на вдосконалення своїх навичок та знань
2. **Комунікація та співпраця.** Компанія прагне створити атмосферу "великої сім'ї", де кожен співробітник відчуває підтримку та взаємоповагу. Вона активно сприяє комунікації між різними відділами та рівнями керівництва, що сприяє створенню сприятливого середовища для ідей та обміну досвідом
3. **Гнучкість та ініціатива.** Компанія підтримує ініціативи своїх працівників, заохочуючи їх до винесення пропозицій та ідей щодо покращення процесів

роботи та прийняття проєктних рішень. Вона стимулює та надає можливість для самостійного планування робочого часу, щоб забезпечити комфортні умови праці для кожного співробітника

Ці принципи формують основу для успішної та ефективної діяльності компанії, допомагаючи створити продуктивну та сприятливу робочу обстановку для всіх працівників.

2.2 Аналіз системи проєктного управління на підприємстві

Компанія ТОВ «Платмастер» є розробником програмних продуктів для загального користування, проте із орієнтацією на бізнес як український так і зарубіжний. Продукт, що наразі знаходиться на стадії бета-тестування для доопрацювання та випуску першої версії, є веб-додатком що представляє собою відкритий торговий майданчик для фізичних осіб, підприємців як фізичних так юридичних осіб, з додатковим функціоналом для ведення документообігу, складських процесів, процесів постачання та вбудованими чатами як каналом комунікації між користувачами. Оскільки основна ідея даного продукту це поєднання зручного та розумного функціоналу для торгування із додатковими супутніми можливостями, а саме замовлення є зовнішньою розробкою, то, для можливості швидкої зміни концепції чи набору актуального функціоналу що буде розроблятися, було прийнято рішення використання підходу гнучкої розробки продукту (Agile), а саме метод Скрам (Scrum).

Скрам є одним з найпоширеніших методів в гнучкому підході до розробки програмних продуктів, цей фреймворк простий і зрозумілий, але за простотою приховані жорсткі вимоги до дотримання процесу та характеристик команди [11]. Оскільки Скрам метод є фреймворком, тобто каркасом, для створення процесу розробки, перелік правил та прийомів що в ньому застосовуються є більш

загальним та надає можливість підбирати та використовувати необхідні елементи саме так, як того вимагає специфіка продукту що розробляється, а також процесів та проєктної команди. В даному випадку для організації ТОВ «Платмастер» такий метод проєктного управління підходить якнайкраще, оскільки проєктна команда що працює над цим продуктом включає певну кількість недосвідчених співробітників, склад проєктної команди може динамічно змінюватись, концепція продукту та необхідний функціонал може коригуватись у відповідності до змін на ринку, а замовник потребує регулярного звітування щодо стану продукту для впевненості в тому що його бачення не розходиться із реалізацією.

На проєкті по розробці продукту Platmix працюють такі члени проєктної команди:

1. Проєктний менеджер;
2. Два старших аналітики вимог;
3. Молодший аналітик вимог;
4. Лідер команди фронтенд розробки;
5. Лідер команди бекенд розробки;
6. Чотири молодших фронтенд розробника;
7. Один молодший бекенд розробник;
8. Молодший дизайнер;
9. Молодший тестувальник;
- 10.DevOps інженер.

Варто детальніше описати залученість членів проєктної команди в процеси розробки продукту.

Проєктного менеджера було залучено до планування роботи над проєктом від самого зародження концепції системи, його рішення безпосередньо впливають на планування можливих витрат ресурсів та строків постачання функціоналу, тому можна вважати даного члена команди однією із найбільш зацікавлених осіб на проєкті. Менеджер на даному проєкті за методом Скрам має роль Скрам майстра (Scrum Master), це означає що на його плечах лежить відповідальність за

організацію ефективної роботи команди та дотримання принципів Скрам [12]. Менеджер проєкту спілкується з кожним співробітником і бачити весь процес зсередини. Також можна вважати, що він виконує роль Власника продукту (Product Owner) оскільки він несе відповідальність за кінцевий результат, тобто яким у підсумку вийде продукт, хоча роль Власника продукту певним чином конфліктує із роллю Скрам майстра, оскільки Власник продукту не очолює процес, а лише розставляє пріоритети [13]. Проте в даному випадку проєктний менеджер поєднує в собі саме ці дві ролі.

Старші аналітики долучаються до роботи над даним продуктом лише за наявності вимог щодо впровадження великого блоку нового складного функціоналу, та працюють на перспективу розробляючи вимоги для майбутніх версій продукту, тобто всупереч концепції методу Scrum ці члени команди займаються детальним описом вимог для запланованих на майбутнє порцій функціоналу. Це зумовлено специфікою знань в предметних областях та досвідом роботи із системами зі схожим функціоналом, а також видом зайнятості цих співробітників. Старші аналітики виконують ролі членів команди розробки за методом Скрам тому не впливають безпосередньо на розстановку пріоритетів та організацію процесів роботи [14].

На даному проєкті саме **молодший аналітик вимог** є основним аналітиком що займається поточними вимогами до продукту. Виявлення, аналіз, документування та перевірка вимог, а також управління ними з орієнтацією на гнучкий підхід, усе це входить до обов'язків даного спеціаліста. В процесі роботи він набуває нових навичок і досвіду корпоративної розробки та навчається на практиці прийомів розробки та управління вимогами. Молодший аналітик є повноцінним членом команди розробки та частково несе на собі обов'язки ролі Власника продукту через знання в предметній області, розуміння для чого продукт необхідний споживачеві та яка в нього цільова аудиторія, а також через розуміння того як і в якому напрямку працює вся команда.

Лідери команд фронтенд та бекенд розробки є старшими технічними спеціалістами на проєкті. Вони слідкують за виконанням задач їх команд розробки, співпрацюють для вирішення певних задач, а також взаємодіють із проєктним менеджером, аналітиком та замовником для розуміння актуального стану продукту та потреб замовника загалом. Ці співробітники є зацікавленими сторонами та можуть впливати на рішення по проєкту та функціоналу. За Скрам методом вони є членами команди розробки.

Молодші фронтенд та бекенд розробники займаються рядовими задачами на проєкті, співпрацюючи та консультуючись з лідерами команд вони розробляють функціонал визначений на відповідну ітерацію. В процесі роботи вони покращують свої навички спілкування та співпраці, а також набувають цінного досвіду у розробці складних систем. За Скрам методом вони також є членами команди розробки.

Дизайнер та тестувальник також грають ролі членів команди розробки за методом Скрам. Дизайнер займається розробкою прототипів користувацьких інтерфесів та графічним оформленням системи вцілому. На основі таких прототипів замовник може оглянути та затвердити можливий дизайн системи ще до того як він буде реалізований. Тестувальник перевіряє чи правильно працює система та її якість. В його обов'язки входить перевірка відповідності реалізованого функціоналу до описаних вимог та можливих сценаріїв використання, а також перевірка системи за якісними характеристиками.

DevOps інженер на проєкті є людиною що відповідає за автоматизацію процесів розробки, тестування, розгортання та управління програмним забезпеченням. Він забезпечує неперервну інтеграцію та неперервну доставку, налаштовує та підтримує інфраструктуру для розробки, моніторингу та відстеження помилок. За методом Скрам він також є членом команди розробки.

Організація робочого процесу за методом Скрам на даному проєкті включає проведення щоденної зустрічі проєктної команди на 10-20 хвилин, так званий щоденний Скрам (Daily Scrum) або як прийнято називати цю зустріч в компанії

стендап (Stand Up), для обговорення виконаних та поточних задач, а також питань що могли виникнути через труднощі [15]. Далі, протягом дня, члени команди, за необхідності, можуть організовувати зустрічі для вирішення питань, гнучко визначаючи час. Оскільки команда проєкту використовує корпоративний месенджер Slack [16] для комунікації, навіть якщо хтось із членів команди працює віддалено вирішення питань не є проблемою. Щоденні зустрічі також організовуються у Slack щоб усі члени проєктної команди могли долучитись до обговорення.

При розбивці та призначенні задач на ітерації розробки, та загалом для управління проєктом команда користується низкою продуктів Atlassian, одним з яких є Jira Software що є комплексною системою відстеження помилок, призначеною для організації спілкування з користувачами, і для управління проєктами [17]. Відповідно до Скрам методу в цій системі можна організувати резерви (Backlog) продукту та спринту в яких зберігатимуться усі артефакти по продукту та окремому спринту необхідні до виконання. Jira надає можливість створити, пов'язати між собою та додати детальний опис для таких артефактів як – епік (Epic), користувацька історія, задача, підзадача, баг. Всі ці артефакти під час планування спринту заносяться у резерв спринту та відображаються на Скрам дошці (Scrum Board), що є візуальним відображення прогресу протягом циклу розробки та поділена на стовпці із відповідними станами артефактів [18]. Як представлено на рис. 2.2, Скрам дошка проєкту PlatmixSystem має наступні стовпці: «TO DO» (до виконання), «IN PROGRESS» (в процесі), «READY IN DEV» (готовий у середовищі розробки), «READY TO TEST» (готовий до тестування), «IN TEST» (тестується), «PENDING» (в режимі очікування), «DONE» (готово).

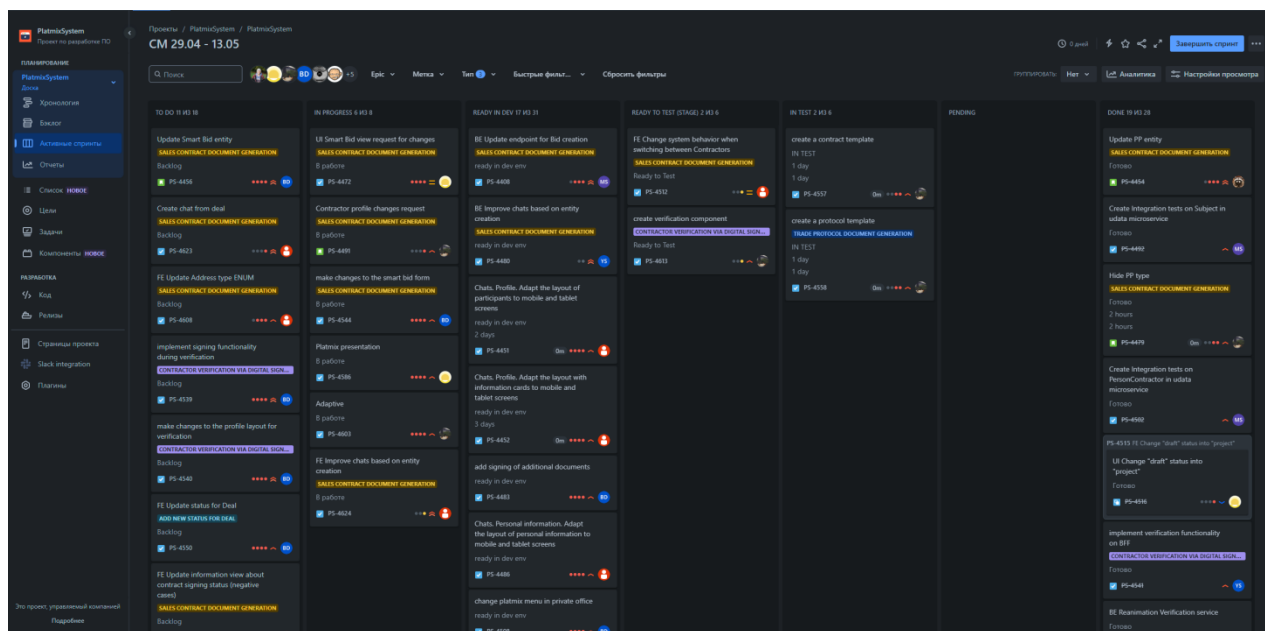


Рисунок 2.2 - Скрам дошка проекту PlatmixSystem

Джерело: розроблено автором в процесі роботи з системою Jira Software

Аналітик вимог під час етапу виявлення вимог разом із проєктним менеджером та замовником визначають пріоритети задач та вирішують які задачі будуть реалізовані у найближчому спринті. Після цього вони заносяться у спринт в стовпчик «TO DO» (до виконання) та призначаються відповідним членам команди для виконання. В процесі розробки лідери команд розробки розбивають деякі задачі на задачі нижчого рівня та призначають виконавців самостійно. Тестувальник за результатами своєї роботи також керує деякими задачами та заводить задачі із помилками – баги, для їх подальшого усунення.

Також за допомогою Jira Software проєктний менеджер слідкує за внесенням відповідних артефактів у певні версії продукту, так звані релізи (Release) – визначені версії продукту в яких відібраний та реалізований певний функціонал у відповідності до пріоритетів. Це допомагає слідкувати за загальним обсягом робіт що потрібно виконати для випуску певної версії продукту [19].

Для роботи над продуктом Platmix, у відповідності до методу Scrum, було вирішено проводити доставку результатів в кінці кожного короткого відрізка розробки - спринта (Sprint), на який відводиться 2 тижні [20]. Доставка результатів

презентується на зустрічі із замовником та зацікавленими особами у вигляді демонстрації доданого в систему функціоналу та його роботи.

Документування вимог, концепції продукту та технічних аспектів також відбувається в одному із продуктів Atlassian, а саме – Confluence. Це корпоративна вікі-система для внутрішнього використання, яку компанії використовують із ціллю створення єдиної бази знань [21]. Дана система має безліч інструментів для документування, управління версіями та управління задачами. Вона інтегрована із Jira Software що надає можливість створювати задачі прямо із тексту цифрових документів, відслідковувати стан задач та пов'язувати задачі із документами. Документація поділена за проєктами та темами, для технічної документації якою користується уся проєктна команда виділений окремий розділ. Загалом для кожного продукту компанії створені окремі розділи в яких можуть працювати співробітники із відповідними правами доступу. Важливо підтримувати актуальність документації оскільки це допомагає не лише наявним робітникам простіше знаходити необхідну інформацію, але й новим робітникам швидше ознайомитись та розібратись у специфіці продукту та процесів розробки.

2.3 Аналіз процесів розробки та управління вимогами до програмних продуктів

Розробка та управління вимогами на проєктах розробки програмних продуктів є однією з найважливіших складових, тому до цього процесу потрібно підходити відповідально. В компанії ТОВ «Платмастер» використовується система Confluence для ведення документації тому саме ця програма є основним інструментом аналітика.

При ознайомленні із документацію аналітиків попередників виявилось що від початку написання документації не було визначено стандартизованих шаблонів документації тому помітним недоліком є різнорідність та деяка хаотичність подання записів. Також графічні представлення не були переведені до єдиного формату через що з'являється складність в читанні та розумінні деяких діаграм. Також в документації зустрічається змішання технічних та користувацьких специфікацій та вимог що певним чином перевантажує документи та сповільнює процес ознайомлення із ними. Через складність та насиченість вимог щодо функціоналу продукту, помітно те, що деякий функціонал не був деталізований, а лише записаний для повторного виявлення в майбутньому. Оскільки із попередніх аналітиків на проєкті нікого в команді не залишилось, розуміння та інтерпретація побажань замовника можуть викривлятися, що призводить до повторного виявлення вимог за певними напрямками і як результат до додаткових витрат часу аналітика та замовника.

Неоднозначні та суперечливі вимоги описані у існуючій документації також сповільнюють процес роботи аналітика вимог через додаткові зустрічі для уточнень, це також негативно впливає на сприйняття процесу роботи аналітика замовником, що в майбутньому може призвести до непорозумінь між зацікавленими сторонами. Через зміну попередніх спеціалістів та неузгодженість щодо формату написання, в документації наявна неконсистентність через що може виникати плутанина та відмінне сприйняття різними членами проєктної команди. Присвоєння унікальних ідентифікаторів різним артефактам в документації, таким як – користувацькі історії, бізнес-правила, обмеження, прототипи та інші, є одним із рекомендованих прийомів по документуванню вимог, на жаль такий прийом не був використаний у наявній документації по проєкту тому до інших проблем додається складність ідентифікації певних артефактів, а також неможливість організації інфраструктури в документації.

Ще одна проблема пов'язана із структурою наявної документації полягала в перевантаженні сторінки документа інформацією по кільком вимогам. Важлива

складова Скрам методу та гнучкого підходу загалом це орієнтація на швидке постачання результатів, або як в ситуації із аналітиком – документації. Тобто окрім того що аналітик вимог повинен швидко постачати документацію, документація повинна мати певний рівень деталізації, не надто глибокий, щоб команда розробки могла швидко ознайомлюватись із нею. В цьому як раз і полягає суть проблеми перевантаження документу, при вивченні документації розробник має багато зайвої інформації яку він або пропустить або забуде, і як результат старання та час витрачений на створення такого документу зводяться до нуля. Актуалізація документації у відповідності до поточного стану продукту також є важливим завданням аналітика, особливо на проєктах гнучкої розробки, тому за відсутності спеціаліста в команді деякий час, актуальність документації втрачається що призводить до конфліктів у співпраці нових та наявних членів команди.

У зв'язку із тим, що на проєкті PlatmixSystem було вирішено застосувати гнучкий підхід до розробки за методом Скрам, процес роботи аналітика вимог та рівень їх деталізації був відкоригований відповідно. На етапі випробувального періоду молодшому аналітику було поставлено завдання ознайомитись із специфікою продукту та предметною областю, а також ставились задачі щодо написання сценаріїв використання для певного функціоналу. У відповідності до формату написання аналітики старшим аналітиком на проєкті, були створені та використані шаблони для сценаріїв використання, бізнес-правил, користувацьких історій, системних сценаріїв використання та користувацьких інтерфейсів. Приклад шаблону сценарію використання можна побачити на рис. 2.3. Оскільки на той момент часу в команді аналітиків налічувалось лише дві людини ініціатива використання шаблонів була запропонована молодшим аналітиком для збереження однорідності документації.

Use Case: Назва кейсу	
ID:	UC-1
Description:	Опис кейсу
Actors:	Користувач, Система.
Preconditions:	• Передумови
Basic Course:	1. Use Case починається коли Користувач ... 2.
Postconditions:	• Післяумови
Business Rules	Бізнес вимоги, обмеження
Related Requirements	Пов'язані вимоги

Рисунок 2.3 - Приклад шаблону сценарію використання

Джерело: розроблено автором в процесі роботи з системою Confluence

Перші зустрічі із замовником проходили в присутності старших колег та старшого аналітика. Під час зустрічей замовник озвучував свої побажання та бачення щодо відрізка функціоналу який він хоче додати в найближчій ітерації, обговорені вимоги документувались у письмовому вигляді, також були поставлені уточнюючі запитання в границях теми зустрічі. Досвід та бачення старшого аналітика допомогли вирішити спірні питання, а також обговорити приховані моменти, що виникли в процесі виявлення вимог.

Наступним етапом після виявлення є аналіз вимог, що є достатньо кропітким процесом. Під час аналізу зібраних вимог було виявлено ряд додаткових питань щодо можливих дій користувача в різних ситуаціях та додаткових вимог що стали супутніми до зібраних, також були проаналізовані залежності описаного функціоналу у вже існуючій системі.

Після проведення додаткової зустрічі із замовником, у відповідності до запропонованих шаблонів було проведено документування вимог. Оскільки на момент їх написання шаблони для високорівневих вимог не використовувались старшим аналітиком через специфіку проєкту на якому він працював, були створені та запропоновані формати шаблонів для опису високорівневих користувацьких вимог та бізнес-концепцій функціоналу. Усі вимоги були атомарно рознесені по

сторінкам деревовидної структури що пропонує система Confluence, як видно на рис. 2.3, та пронумеровано унікальними ідентифікаторами для легшого пошуку. Також усі відповідні вимоги були пов'язані між собою для збереження консистентності та можливості відслідковування зв'язків. В процесі роботи страший аналітик регулярно проводив рецензію написаних специфікацій, після чого узгоджувались певні зміни в структурі документації та написаних вимогах задля покращення вихідного результату.

- ▼ GSFP. EDocuments Management
 - ▼ GSFP. EDM. Business Context
 - GSFP. EDM. BC. General Requirements Management
 - GSFP. EDM. BC. Generate EDocuments
 - ▼ GSFP. EDM. BC. Custom Contract
 - ▼ GSFP. EDM. BC. CC. Requirements
 - GSFP. EDM. BC. CC. R. User Requirements
 - GSFP. EDM. BC. CC. R. Use Cases

Рисунок 2.4 – Деревовидна структура сторінок Confluence

Джерело: розроблено автором в процесі роботи з системою Confluence

Заключним етапом після документування є етап перевірки та погодження вимог із замовником, проте в силу того що проєктна команда та менеджер проєкту прагнули пришвидшити процес розробки, а замовник не був ознайомлений із прийомами та форматом документування цей етап опускався, що значить те що документація напряду надходила до розробників для ознайомлення. Це, звісно, створює додаткові ризики та може впливати у те що під час демонстрації реалізації даного функціоналу замовник висловить незадоволеність в отриманому результаті через розбіжність реалізації та його бачення.

Управління вимогами є процесом супутнім із життєвим циклом вимог і включає аналіз та внесення змін, а також, зазвичай, вимагає регламентованого

процесу управління вимогами заради зменшення негативного впливу змін та підготовки до їх впровадження. Оскільки на описаному проєкті використовується гнучкий підхід до розробки, внесення змін в існуючі вимоги є регулярним завданням. Через відсутність ідентифікаторів та добре розподіленої структури завдання управління вимогами значно ускладнюється. За відсутності регламентованого процесу управління вимогами доводиться звертатись до старших колег за порадою або ж придумувати рішення на місці. В першому випадку це призводить до створення окремих шаблонів для опису інкрементів змін, в другому випадку, в залежності від обставин, рішеннями можуть бути редагування старої документації та використання інструментів виділення тексту задля фіксації нових змін, або створення нових версій документу із внесеними змінами. Інструмент Confluence пропонує зручний механізм роботи із версіями що дозволяє окремо оглянути контент що був змінений в попередній версії документу.

2.4 Модель управління вимогами до програмних продуктів, прийнята в компанії

Для представлення моделі розробки та управління вимогами в ТОВ «Платмастер» було прийнято рішення використати два варіанти графічного представлення, а саме:

- Діаграма діяльності (Activity Diagram);
- BPMN діаграма (Business Process Model and Notation Diagram).

Діаграми діяльності разом із діаграмами варіантів використання та кінцевих автоматів вважаються діаграмами поведінки, оскільки вони описують те, що має статися в системі, що моделюється. Зацікавленим сторонам потрібно вирішити багато питань, тому важливо спілкуватися чітко та лаконічно. Діаграми діяльності

допомагають людям, які займаються бізнесом і людям які займаються розробкою, об'єднатися, для розуміння процесу та поведінки системи [22]. Іншими словами ця проста діаграма верхнього рівня покроково розкриває суть певної діяльності що відбувається в рамках певної системи. Така діаграма є зручним та швидким засобом графічного представлення покрокових подій під час діяльності певного актора. Прикладом може бути діяльність користувача в банківській системі, де користувач послідовно виконує такі дії:

1. Перевірка балансу;
2. Списання або нарахування коштів на рахунок;
 - а. Списання неможливе
 - б. Списання можливе;
3. Оновлення балансу.

У свою чергу Business Process Model and Notation діаграма також відома як діаграма BPMN, є корисним інструментом, який відображає внутрішній бізнес-процес. Мета діаграми BPMN — презентувати візуальне представлення певної бізнес-процедури, щоб вона була зрозуміла всім зацікавленим сторонам. Таким чином різні бізнес-підрозділи можуть бачити співпрацю та транзакції, які відбуваються в організації [23]. Окрім користі для бізнес-підрозділів дана діаграма може спростити роботу команди розробки якщо за її допомогою описати процес що відбувається в рамках системи. Якщо як учасників додати різні компоненти системи, а як сам процес описати їх взаємодію, за допомогою такої нотації можна створити покроковий опис роботи сервісів системи в рамках певного процесу. Наприклад таким чином можна описати те як зовнішня інтегрована система взаємодіє із системою що розробляється, в рамках сценарія використання, які завдання виконують обидві системи в процесі, які події відбуваються та як розгалуження в процесі можуть вплинути на результат.

Нижче представлена діаграма діяльності яка описує покрокову діяльність в рамках процесу розробки вимог «як є», в ТОВ «Платмастер», рис. 2.5.

Як видно на рис. 2.5, стартовою точкою цієї діаграми є ініціація проєкту, хоча слід зазначити що необов'язково саме це повинно бути відправною точкою оскільки на проєкті PlatmixSystem використовується гнучкий підхід до розробки, а саме метод Скрам, стартовою точкою може бути ініціація зустрічі для виявлення вимог для найближчого відрізка розробки, що не має значного впливу на процес загалом, проте тягне за собою необхідність аналізу впливу нових вимог на вже існуючу документацію та впроваджений функціонал системи.

Першою діяльністю є виявлення вимог, що включає роботу із замовником та зацікавленими сторонами задля отримання вичерпного набору вимог щодо функціоналу продукту. Далі проходить фіксація результатів зустрічі, оскільки за результатами зустрічі може бути виявлений чималий список вимог різних рівнів усі обговорені вимоги та теми фіксуються в документації, в документ із записами про зустрічі команди, в текстовому вигляді. Такий документ корисний при виникненні конфліктів та непорозумінь в команді щодо певного функціоналу та рішень, в таких випадках можна звернутись до цього документу для вирішення питань та конфліктів, оскільки він має зафіксовані прийняті рішення та результати обговорень.

Наступною на діаграмі йде умова того чи всі вимоги було виявлено, це передбачено на випадок того що за час зустрічі зацікавлені сторони не встигли обговорити або надати усі вимоги. Таким чином якщо було виявлено не усі вимоги процес повертається до діяльності виявлення вимог.

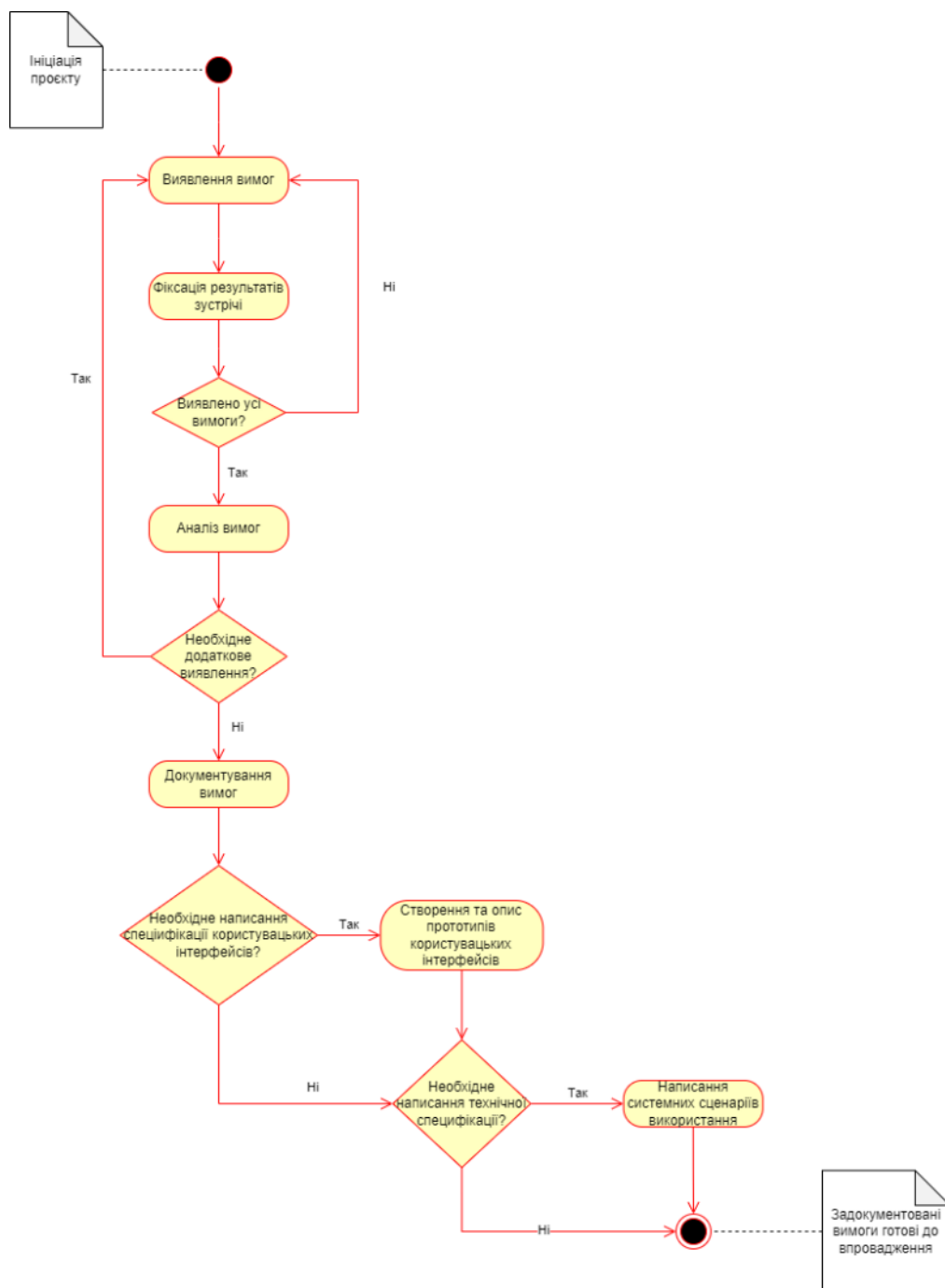


Рисунок 2.5 – Діаграма діяльності – Розробка вимог «як є»

Джерело: розроблено автором

Якщо усі вимоги було виявлено процес переходить до діяльності аналізу вимог, під час якого аналітик вимог аналізує виявлені вимоги та, за необхідності, фіксує додаткові питання та моменти що необхідно уточнити. Якщо після аналізу вимог з’являється необхідність додакового виявлення або уточнення процес із цієї умови повертається на діяльність виявлення вимог.

Якщо в додатковому виявленні вимог немає потреби процес переходить до діяльності документування вимог, під час якої аналітик формалізує зібрані вимоги в документації по продукту.

Додатковими умовами в процесі розробки вимог можуть бути написання специфікації користувацьких інтерфейсів та написання технічної специфікації, а саме системних сценаріїв використання. Якщо від команди розробки надходить запит на деталізацію вимог щодо користувацького інтерфейсу, аналітик вимог створює базові прототипи інтерфейсів та описує їх поведінку в специфікації користувацьких інтерфейсів, після чого дизайнер на їх основі створює повноцінні прототипи та оптимізує їх. Якщо від команди розробки надходить запит на деталізацію технічних вимог, а саме опис поведінки системи на стороні серверу для прискорення та спрощення роботи команди бекенд, аналітик описує системні сценарії використання де акторами є компоненти системи і поетапно показана взаємодія таких компонентів в рамках певної діяльності користувача в системі. Такий вид специфікації хоч і є складним в написанні, проте значно пришвидшує роботу проєктної команди.

Як результат діяльності по розробці вимог отримано набір задокументованих вимог що готові до впровадження проєктною командою. Діаграма діяльності в цьому випадку описує процес розробки вимог в ТОВ «Платмастер» достатньо поверхнево, проте цього вистачає для формування загального розуміння щодо того як така діяльність відбувається на проєкті PlatmixSystem даної компанії. Орієнтація на метод гнучкої розробки Скрам впливає на дану діяльність не найкращим чином, оскільки в процесі розробки вимог опускається один з найважливіших етапів – перевірка вимог. Через це задокументовані вимоги, а в майбутньому і впроваджений функціонал, можуть не відповідати баченню замовника та потребам користувачів, що може призвести до внесення значних змін у вимоги, і як результат до критичних простоїв та наднормових перевитрат при їх внесенні.

із замовником. Як видно при переході процесу в пул інших зацікавлених сторін за оператором АБО йде визначення необхідності коригування вимог, якщо вносити зміни непотрібно процес в цьому пулі завершується, якщо необхідно внести зміни зацікавлені сторони висувають пропозиції щодо коригування вимог, якщо замовник згоден зі змінами процес результує у внесення змін у вимоги, а якщо ні, то процес в цьому пулі завершується. Варто зазначити що іншими зацікавленими сторонами в цьому випадку є члени проєктної команди та топ менеджери компанії тому вплив на зібрані вимоги зі сторони відсутній, кожен учасник зустрічі знайомий із контекстом та продуктом, саме тому вони можуть пропонувати зміни в ході виявлення вимог.

Паралельне відгалуження процесу, що переходить до завдання замовника, яке полягає в тому що він висуває вимоги та побажання, після цього переходить до пулу аналітика вимог, який записує вимоги замовника у текстовому та аудіо форматах. Якщо аналітику в процесі виявлення вимог необхідні уточнення щодо функціоналу він задає питання замовнику, а той відповідає на них, результатом чого є деталізація вимог до необхідного рівня. Наступним на діаграмі є оператор І що об'єднує розгалудженні процеси та включає деталізацію вимог замовником та внесення змін зацікавленими сторонами.

Аналітик вимог фіксує вимоги та результати зустрічі після чого замовник висуває своє бачення щодо пріоритетів впровадження вимог. Зазвичай процес пріоритизації відбувається після перевірки вимог проте потік процесу розробки вимог в ТОВ «Платмастер» включає цей етап саме в процесі їх виявлення. Якщо є необхідність залучення зацікавлених сторін до пріоритизації вимог, вони опитуються та пропонують своє бачення або певні корективи у запропоновані замовником пріоритети.

Після визначення пріоритетів завершується етап виявлення вимог та починається етап аналізу вимог. Як зазначено на діаграмі після аналізу вимог аналітик може визначити необхідним додаткове виявлення вимог для уточнення неоднозначностей, вирішення колізій та обговорення додаткових супутніх вимог,

це призводить до повторної ініціації зустрічі із замовником. Цей процес може повторюватись невизначену кількість разів для отримання вичерпного набору однозначних вимог.

Якщо додаткове виявлення вимог не є необхідним, аналітик переходить до завдання документування вимог у відповідних специфікаціях. За необхідності консультації інших зацікавлених сторін аналітик направляє відповідний запит та організує зустріч із ними. Додатково, після консультації із зацікавленими сторонами, якщо члени проєктної команди просять створити базові прототипи та опис до користувацьких інтерфейсів, аналітик виконує це завдання. Схожим чином при запиті від команди бекенд для опису технічної специфікації, аналітик вимог описує детальні системні сценарії використання.

Останнім та заключним етапом є перевірка документації. Оскільки на проєкті PlatmixSystem цей етап розробки вимог опускається, лише сам аналітик перевіряє свою роботу, що є значним недоліком як і описувалось раніше. При необхідності редагування та виправлення помилок аналітик вносить зміни в написану документацію.

Результатом процесу є описані специфікації вимог що готові до ознайомлення та впровадження проєктною командою.

Процес управління вимогами «як є», зображений у вигляді діаграми діяльності, наведений нижче рис. 2.7.

Оскільки при управлінні вимогами початковою точкою є внесення певних змін в існуючі вимоги то на діаграмі стартовою точкою представлено ініціацію внесення змін. Діяльність збору змін майже не відрізняється від діяльності виявлення вимог оскільки так само включає взаємодію із замовником та зацікавленими сторонами. Фіксація результатів зустрічі також відповідає такій діяльності як на діаграмі – рис. 2.5, та включає фіксацію виявлених змін.

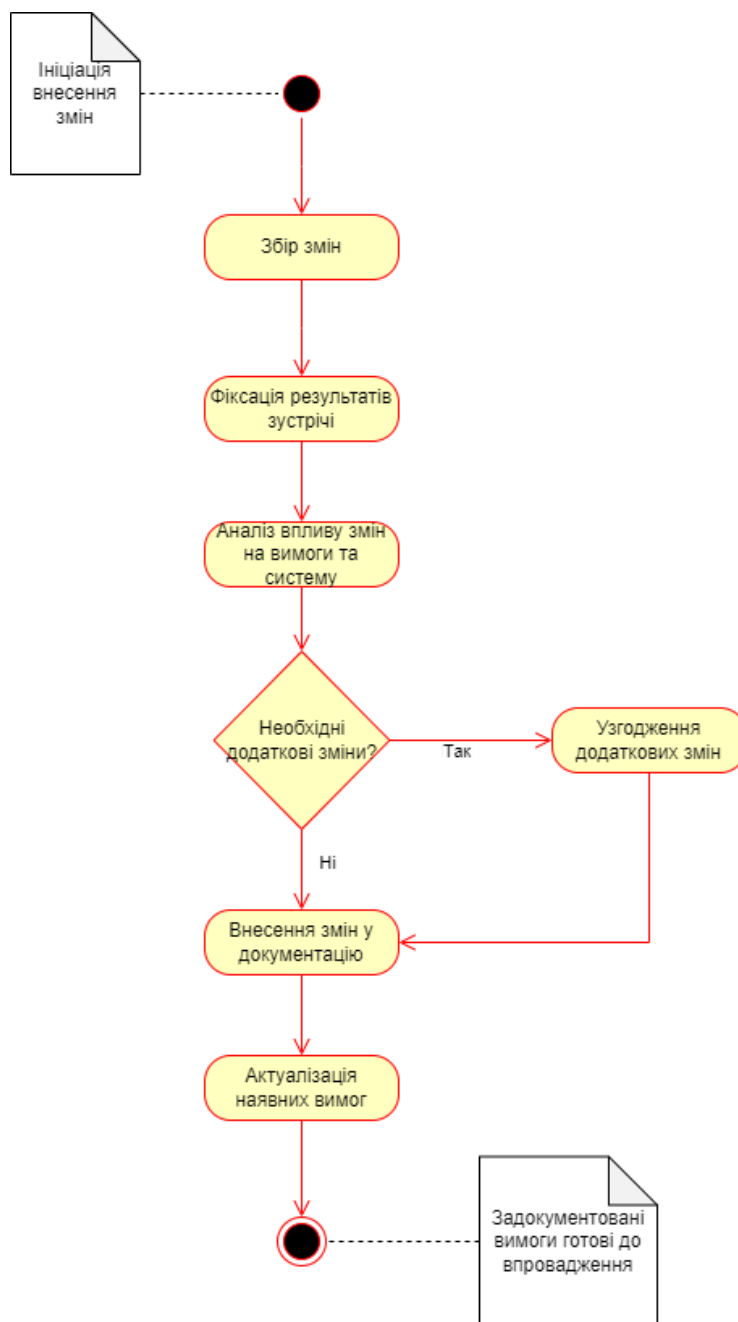


Рисунок 2.7 - Діаграма діяльності – Управління вимогами «як є»

Джерело: розроблено автором

На даній діаграмі помітно те що відсутній повторюваний цикл із умовою визначення необхідності додаткового виявлення вимог, це пов'язано із тим що зазвичай обсяг змін є незначним через специфіку гнучкого підходу до розробки і як результат проведення додаткових зустрічей не є необхідним. Тому наступним етапом є проведення діяльності по аналізу впливу змін на вимоги та систему, що є кропітким та дуже важливим завданням аналітика вимог. Можна вважати цю діяльність ключовою у впровадженні будь-яких змін, оскільки вона надає

можливість виявити приховані залежності що можуть спровокувати виявлення додаткових вимог. Як результат аналізу впливу, процес може переходити до умови впровадження додаткових змін. Якщо була виявлена необхідність впровадження додаткових змін аналітик виконує діяльність по узгодженню змін із зацікавленими сторонами.

Останніми двома діяльностями є внесення змін в документацію, а також додаткова перевірка та актуалізація наявних вимог для збереження відповідності після внесення змін. Результатом діяльності по управлінню вимогами є, як і у випадку із розробкою, набір задокументованих вимог що готові до впровадження проєктною командою. Внесені зміни були проаналізовані та впроваджені, за необхідності, також були внесені додаткові зміни після аналізу впливу.

Розглянемо більш детальну діаграму бізнес-процесу управління вимогами «як є» – рис. 2.8, що наведена нижче.

Як видно на цій діаграмі бізнес-процес управління вимогами певним чином відрізняється від процесу розробки вимог, проте поєднує в собі використання тих самих трьох пулів учасників – аналітик вимог, замовник, інші зацікавлені сторони.

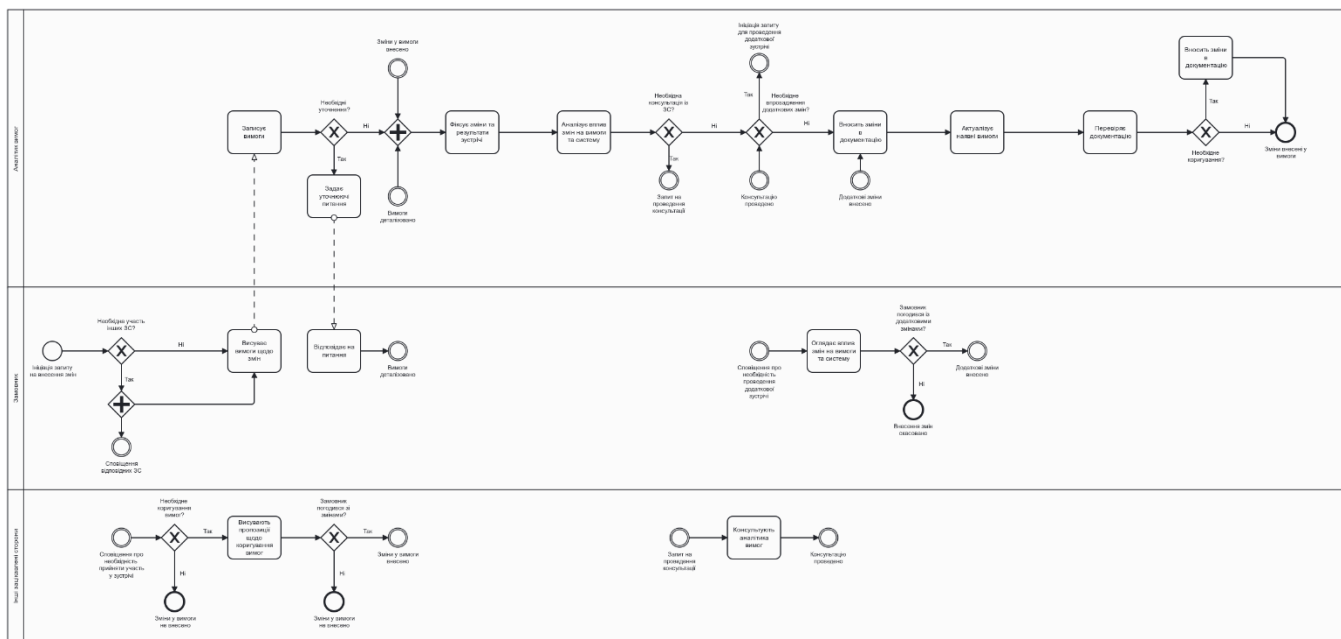


Рисунок 2.8 - BPMN діаграма – Управління вимогами «як є»

Джерело: розроблено автором

Початковою подією, схожим чином як на діаграмі рис. 2.8, є ініціація запиту на внесення змін від замовника. В ТОВ «Платмастер» на проєкті PlatmixSystem в основному лише замовник ініціює зміни у вимогах, оскільки є основним їх постачальником. Відповідно до того як це відбувається при розробці вимог, можуть бути залучені інші зацікавлені сторони, це відбувається через оператор АБО зображений далі на діаграмі. Тому видно що процес збору змін, що завершується на завданні аналітика щодо фіксації змін та результатів зустрічі нічим не відрізняється від процесу збору вимог який зображено на діаграмі – рис. 2.8. Так само замовник висуває свої вимоги щодо змін, а зацікавлені сторони, за умови їх участі, можуть коригувати ці зміни, аналітик може ставити уточнюючі запитання задля деталізації змін. Звісно єдиною відмінністю є те що обробляються саме зміни до вимог, а не початкові вимоги.

З етапу фіксації змін та результатів зустрічі, процес управління вимогами переходить до етапу аналізу впливу змін на вимоги та систему. Як зазначалось раніше цей етап є найважливішим при управлінні вимогами, оскільки надає можливість виявити додаткові вимоги та залежності між ними. Після цього аналітик визначає чи є необхідним проведення додаткових зустрічей із зацікавленими сторонами та замовником. Якщо аналітику необхідна додаткова консультація від проєктної команди він організовує відповідну зустріч для вирішення своїх питань. Якщо були виявлені супутні вимоги і, як результат, необхідна додаткова зустріч із замовником, процес переходить в пул замовника де той виконує завдання огляду впливу змін та обирає чи затвердити додаткові зміни. Якщо замовник затверджує їх, процес повертається в пул аналітика вимог та переходить до виконання завдання внесення змін в документацію. Якщо замовник не згоден зі змінами процес повністю завершується скасуванням внесення зібраних змін.

Після того як аналітик вносить зміни в документацію, він також перевіряє та актуалізує наявні вимоги для збереження відповідності, як зазначалось раніше.

Заключним етапом є перевірка аналітиком внесених змін та коригування документації, за необхідності.

В результаті бізнес-процесу управління вимогами були зібрані, проаналізовані та затверджені зміни, а також актуалізовані наявні вимоги. Або як альтернативний варіант завершення процесу замовник скасував внесення змін тому документація залишилась незміненою.

Підсумовуючи результати моделювання процесів розробки та управління вимогами в ТОВ «Платмастер» за допомогою діаграм діяльності та BPMN діаграм можна зазначити що замовник є найбільш впливовою зацікавленою стороною, оскільки саме він зазвичай ініціює зустрічі для виявлення вимог або впровадження змін. Інші зацікавлені сторони також мають вплив на вимоги та зміни, проте не мають такої влади як замовник, а тому не можуть ініціювати виявлення нових вимог або впровадження значних змін, оскільки навіть за їх участі замовник не завжди погоджується щось змінювати у своїх вимогах. Ситуація розгортається подібним чином і на етапі пріоритезації вимог який в даному випадку включено до процесу виявлення вимог, зацікавлені сторони мають вплив на пріоритети вимог лише в тому випадку коли замовник вважає що їх варто опитувати для коригування. Одним з найбільших недоліків описаного процесу розробки вимог є відсутність перевірки вимог замовником та зацікавленими сторонами, що, як було зазначено раніше, може негативно вплинути на реалізації функціоналу, оскільки описані вимоги можуть йти у розріз бачення замовника. Щодо процесу управління вимогами слід зауважити, що відсутній повторюваний цикл, як в процесі розробки, а також основною відмінністю є саме артефакт над яким проводяться дії – зміни до вимог. Увагу привертають етапи аналізу впливу змін та узгодження додаткових змін, оскільки це два найважливіші кроки в роботі зі змінами, вони дозволяють виявити додаткові, проте необхідні, зміни і узгодити їх із замовником. Результатами обох процесів є достатня та готова для реалізації документація по продукту.

РОЗДІЛ 3

ПРОПОЗИЦІЇ ЩОДО УДОСКОНАЛЕННЯ

ПРОЦЕСІВ РОЗРОБКИ ТА УПРАВЛІННЯ ВИМОГАМИ В ТОВ

«ПЛАТМАСТЕР»

3.1 Систематизація та структуризація процесу розробки вимог

Аналіз процесу розробки вимог в ТОВ «Платмастер» (див п. 2.3) показав, що в компанії використовується гнучкий підхід до розробки програмного продукту за методом Скрам. Але питанню розробки й управління вимогами приділяється недостатньо уваги, зокрема, в існуючій документації по проєкту, вимоги описуються поверхнево, орієнтуючись на швидку доставку результатів.

Крім того, оскільки не розроблено чіткого протоколу/регламенту цього процесу, як результат, має місце така ситуація: зацікавлені сторони, що раніше не працювали з вимогами або не співпрацювали із аналітиком вимог, не розуміють суті та етапів роботи з вимогами, при цьому вони мають лише поверхнєве уявлення того чим займається аналітик і яку користь проєкту він приносить, і чи приносить її взагалі.

Замовник та зацікавлені сторони мають розуміти, чим займається аналітик вимог у проєкті та які етапи включає в себе сам процес розробки вимог. Тому в регламенті слід чітко описати етапи процесу розробки вимог, включивши до нього усі базові елементи, що були описані в п. 1.2, а саме:

1. *Виявлення (elicitation);*
2. *Аналіз (analysis);*
3. *Документування (specification);*
4. *Узгодження (validation).*

Для формування регламенту процесу розробки вимог важливо проводити зустрічі з іншими колегами з аналітичного відділу, оскільки більш досвідчені аналітики можуть допомогти формалізувати та покращити базовий процес розробки. Доцільно також організувати зустріч із замовником, проєктною командою та зацікавленими сторонами та обговорити з ними результати опису кожного з цих елементів, пояснюючи суть кожного із них та відповідаючи на питання, що можуть виникнути. Якщо під час презентації регламенту зацікавлені сторони матимуть пропозиції, їх варто розглянути та обговорити на місці. Після внесення змін, якщо такі були запропоновані та ухвалені, отримуємо узгоджений документ, що регламентує процес розробки вимог, який у подальшому стане відносним внутрішнім стандартом цього процесу в майбутніх проєктах компанії.

Метою створення та узгодження регламенту процесу розробки вимог в компанії ТОВ «Платмастер» є розкрити інвесторам і стейкхолдерам (зацікавленим сторонам) суть роботи аналітика та зробити сам процес розробки вимог прозорим. Оскільки в даній компанії в процесі розробки вимог етап їх узгодження опущено (див. п. 2.4), саме його варто додати як обов'язкову ланку для усунення потенційних ризиків простою активів через невідповідність вимог, що сформульовані, баченню замовника. Через те, що в процес визначення вимог буде доданий елемент їх узгодження, усі зацікавлені сторони та члени проєктної команди також отримають важливі знання щодо того як саме відбуватиметься процес розробки вимог у їх проєкті.

Розглянемо детальніше кожний із елементів процесу розробки вимог.

Першим етапом є **виявлення вимог**. В п. 1.2 описано рекомендовані прийоми та формати проведення зустрічей із замовником та зацікавленими сторонами на цьому етапі. Зазначимо, що в ТОВ «Платмастер» на цьому етапі в проєктах використовується прийом, що має назву **визначення концепції продукту та границь проєкту**, за якого документ про концепцію та границі створюється ще на початку зародження (на етапі ініціації) проєкту. Він є базовою специфікацією при запуску нового проєкту, тому вносити зміни у порядок проведення цього етапу

не варто. Єдине що варто поліпшити, це усунути неоднорідність та довільність подання інформації, оскільки це ускладнює ознайомлення із документацією та може призвести до створення подібних документів із абсолютно іншою структурою. Варіантом розв'язання цієї проблеми є впровадження гнучкого шаблону такого документу, який аналітики зможуть використовувати в майбутньому. Під гнучким шаблоном мається на увазі можливість коригування документу в залежності від специфіки концепту, при цьому дотримуючись певного незмінного каркасу шаблону.

Такі прийоми як **визначення класів користувачів, виокремлення з користувачів затятих прибічників продукту, створення фокус-груп, спостереження за користувачами на робочих місцях та роздача опитувальних листів** усі орієнтовані на співпрацю із потенційними та безпосередніми користувачами системи. На жаль жоден із цих прийомів не використовується при виявленні вимог в ТОВ «Платмастер». Зазвичай замовник виступає як представник користувачів і висуває відповідні вимоги. Нині компанія не має ресурсів для організації набору користувачів для співпраці заради виявлення вимог та проведення заходів із фокус-групами. Проте в майбутньому рекомендується використати бета-версію продукту для публічного тестування з метою отримання зворотнього зв'язку від користувачів. Окрім цього необхідним є визначення аналітиком класів користувачів, навіть за відсутності можливості співпраці із представниками кожного із класів. Аналітику вимог варто зайнятись виокремленням класів користувачів задля детальнішого виявлення вимог, в процесі виявлення замовник та зацікавлені сторони можуть висувати свої пропозиції щодо вимог, які базуються на можливих потребах того чи іншого класу користувачів. Оскільки продукт що розробляється, матиме широкий спектр користувачів зі сфери торгівлі, співпраця із користувачами є неминучою, а тому правильним рішенням буде застосування цих рекомендованих прийомів на етапі виявлення вимог до нової версії продукту задля збільшення його конкурентоспроможності.

Визначення користувацьких вимог та визначення системних подій та реакцій на них, обидва ці прийоми успішно використовуються в компанії ТОВ «Платмастер» у проєкті PlatmixSystem. Користувацькі вимоги є мінімально необхідним рівнем деталізації вимог у проєктах в даній компанії, це пов'язано із використанням методу Скрам де при проєктному управлінні створюється резерв проєкту та спринта, які складаються із таких артефактів як користувацькі історії, епіки, задачі та інших, як описано в п. 2.2. Саме тому користувацькі вимоги є першим рівнем вимог що виявляються у проєкті.

Щодо прийому **визначення системних подій та реакцій на них**, то він застосовується на проєкті PlatmixSystem через специфіку концепту продукту та функціоналу що розробляється. Системні події хоч і фіксуються проте записуються на рівні користувацьких вимог як їх частина, що є вкрай неправильним підходом, оскільки такі події скоріше відносяться до рівня системних вимог. Тому логічною є пропозиція виокремити опис цих подій в окремому документі або розділі документації з метою розмежування подій та реакцій, що відбуваються в системі, від користувацьких вимог.

Прийоми **проведення інтерв'ю та семінарів для виявлення вимог** також постійно використовуються в ТОВ «Платмастер» оскільки саме ці зустрічі є основним джерелом вимог. Досвідчений аналітик вимог при проведенні таких зустрічей, зазвичай бере на себе роль організатора та ведучого/фасилітатора. Необхідно вміти правильно та з повагою спілкуватись із замовником та зацікавленими сторонами, ставити правильні уточнювальні питання та тримати курс обговорення в рамках теми зустрічі. Оскільки чіткого алгоритму дій на таких зустрічах не існує аналітик повинен сам приймати рішення та проявляти комунікативні навички. Проте у цьому випадку, організації варто регламентувати проведення таких зустрічей та визначити чіткі часові рамки для кожного виду. В залежності від виду варто впровадити такі обмеження:

- Для інтерв'ю – до двох годин із перервою;
- Для семінарів – до трьох годин із перервою;

- Для мозкового штурму – до однієї години.

Таким чином співробітники та замовник зможуть планувати власний час та організовувати робочий процес без простоїв. Оскільки теми зустрічей можуть мати різне наповнення рекомендований та реальний час проведення можуть відрізнятись, проте відповідність цим обмеженням забезпечать продуктивне виявлення вимог.

Наступні практики, що використовуються в процесі виявлення вимог в ТОВ «Платмастер», - **аналіз документів, вивчення звітів про проблеми та повторне використання існуючих вимог**. Щодо першої практики в цьому списку варто зазначити що оптимізація аналізу документів може залежати лише від того як ця документація була описана та які особисті якості роботи з інформацією має аналітик. У свою чергу, вивчення звітів про проблеми потребує наявної системи яку вже використовують користувачі, тобто це означає роботу з наявним набором даних для отримання інформації, а згодом і знань, щодо проблем та помилок з якими стикнулись користувачі. В планах по проєкту PlatmixSystem отримання відгуків та звітів про помилки відведено важливу роль для покращення випущеного продукту. Оскільки ідея екосистеми продуктів та специфіка технічної архітектури покривають можливість створення копій продукту для продажу зовнішнім замовникам, використання прийому із задіянням існуючих вимог також є розповсюдженою практикою.

Другою складовою в процесі розробки вимог є **аналіз вимог**. Розглянемо те які прийоми цієї складової використовуються на проєктах компанії, та як можна їх покращити.

Прийом **моделювання середовища програми** є ефективним способом графічного представлення продукту що використовується в ТОВ «Платмастер» як на початкових етапах розробки продукту так і на більш пізніх етапах розробки. Через те що проєктна команда проводить процес розробки у відповідності до методу Скрам, а створення діаграм та графічних представлень функціоналу та продукту є складним завданням що потребує часу, постає питання ефективності створення таких моделей, оскільки після їх моделювання команда вкрай рідко

користується ними. Проте графічний опис продукту є не лише важливим, а й вкрай необхідним завданням. Це пов'язано з тим що новим членам проєктної команди такі моделі значно спрощують процес сприйняття та ознайомлення із продуктом що розробляється. У випадку проєкту PlatmixSystem рекомендується скоротити кількість моделей що описуються за замовчанням для нового та існуючих продуктів. Наразі список моделей містить діаграму сутність-зв'язок, діаграму сценаріїв використання, BPMN діаграму, діаграму класів, діаграму послідовності та діаграму архітектури системи. З огляду на те які діаграми частіше застосовуються проєктною командою та аналітиком безпосередньо, задля економії часу та запобігання виникнення простоїв варто скоротити цей список таким чином:

1. Діаграма сутність зв'язок (Див. Додаток А: рис.А1);
2. Діаграма класів (Див. Додаток А: рис.А2);
3. Діаграма послідовності (Див. Додаток А: рис.А3);
4. Діаграма архітектури системи (Див. Додаток А: рис.А4).

Саме ці чотири діаграми найчастіше використовуються проєктною командою.

Діаграма сценаріїв використання є зручним способом представлення усіх сценаріїв використання та те як вони пов'язані між собою, системою та користувачем, проте наразі необхідності в графічному зображенні цих зв'язків немає через недостачу людських ресурсів в аналітичній команді, через що для аналітика вимог ця задача відпадає як пріоритетна. В майбутньому, однак, створення таких діаграм потрібно повернути оскільки це є необхідним етапом при описі вимог.

BPMN діаграма з описом сценаріїв використання або торгових процедур на проєкті PlatmixSystem є детальним способом представлення усього бізнес-процесу покроково, проте займає величезну кількість часу та використовується виключно аналітиками вимог в їх роботі для швидкої орієнтації в потоці процесу. Така діаграма, звісно, спрощує роботу аналітика та навіть може бути представлена перед зацікавленими сторонами для детального огляду певного бізнес-процесу, проте у

цьому випадку з даною організацією, створення такої діаграми не несе значної користі і створює ризик простою аналітичних ресурсів.

Варто зазначити що узгоджений список графічних моделей не відмінняє можливості того що певний член проєктної команди чи зацікавлена сторона висунуть запит на створення моделі якої в цьому списку немає, в такому випадку аналітик повинен приймати рішення у співпраці із проєктним менеджером, і якщо такий запит є раціональним то його можна виконати.

Наведені вище чотири види діаграм були залишені як обов'язкові через те що діаграма сутність-зв'язок є основним способом представлення зв'язків на різних рівнях як на бізнес-рівні так і на системному рівні. А інші три діаграми є допоміжними до технічної специфікації і допомагають членам команди розробки орієнтуватись на попередньо створену структуру.

Створення прототипів як рекомендований прийом аналізу вимог також використовується в компанії ТОВ «Платмастер». На проєкті PlatmixSystem в цей прийом включається опис та створення базових прототипів користувацьких інтерфесів. Цей відрізок роботи значно сповільнює обробку отриманих вимог, оскільки аналітик сам створює як графічне та і контекстне наповнення вимог до користувацьких інтерфейсів. У зв'язку із використанням методу Скрам цей процес необхідно прискорити та спростити. Варіантом вирішення цього завдання може бути розподіл обов'язків по створенню інтерфейсів та опису їх поведінки між дизайнером та аналітиком. Аналітик вимог буде описувати сценарії використання орієнтуючись на необхідні у візуальному контурі контролю, після чого дизайнер створить необхідні прототипи інтерфейсу. Таким чином аналітик встигатиме покривати вимоги користувацького рівня сценаріями використання, а дизайнер тим часом створить необхідні інтерфейси які можна буде презентувати замовнику. При таких змінах в підході до створення прототипів опускається етап детального опису поведінки інтерфейсів що, як результат, впливає на процес роботи дизайнера, оскільки він сам буде приймати рішення щодо їх поведінки.

Аналіз здійсненності та розстановка пріоритетів вимог як рекомендовані прийоми аналізу вимог застосовуються на проєктах організації ТОВ «Платмастер», проте другий з них має певні модифікації. Майже на кожній зустрічі з виявлення вимог або виявлення змін присутні лідери команди фронтенд та бекенд розробки що вказує на те що аналіз здійсненності вимог проходить одразу під час їх виявлення що значно прискорює роботу аналітика в цьому напрямку. Схожим чином проходить і розстановка пріоритетів вимог, одразу під час зустрічі виявлення. Такий підхід хоча і дозволяє скоротити час що витрачається на ці прийоми, проте він створює можливість для виникнення неоднозначностей та колізій в майбутньому. Пропонується залишити використання цих прийомів на етапі виявлення вимог, проте варто змінити підхід до розстановки пріоритетів таким чином – на етапі виявлення вимог лише попередньо визначати пріоритезацію вимог для того щоб аналітик міг орієнтуватись на ці пріоритети під час проведення аналізу, а після етапу узгодження вимог проводити додаткову пріоритезацію за участі проєктного менеджера. Таким чином якщо після виявлення та аналізу будуть виявлені додаткові вимоги, що вплине на розстановку пріоритетів, пріоритети будуть остаточно узгоджені вже після перевірки всього обсягу задокументованих вимог.

Створення словника даних є достатньо простим в реалізації прийомом його варто впровадити на усіх проєктах в організації задля забезпечення розуміння термінів членами проєктної команди. Це дозволить співробітникам спілкуватись, так би мовити, «однією мовою» та усуне неоднозначності у трактуванні певних термінів.

Третьою складовою розробки вимог є **документування**, що включає формалізацію оброблених вимог у визначених форматах. В компанії ТОВ «Платмастер» використовуються спеціальні програмні засоби для ведення документації, це надає можливість створити унікальну структуру та впровадити певні стандарти у роботу з вимогами.

Першим і одним із найважливіших прийомів є **використання шаблонів специфікацій вимог**. Як зазначалось в п. 2.3 розроблювана документація по проєкту з самого початку не мала певних шаблонів специфікацій, через що формати подання вимог та графічних моделей відрізняються в місцях де працювали різні аналітики. Тому варто впровадити чіткі стандарти шаблонів для різних специфікацій та об'єднати їх у відповідному документі. Альтернативним рішенням може бути використання шаблонів самого інструменту Confluence в якому ведеться документація, проте, на жаль, наявні шаблони не покривають усіх потреб при розробці вимог. На основі структури вимог які описує старший аналітик повинні бути створені шаблони для кожного із артефактів, та винесені в окремий розділ звідки їх можна буде використовувати. Таким чином можна забезпечити однорідність документації, збереження її структури, однозначне сприйняття та відповідність певним стандартам якості вимог.

Інструмент Confluence дозволяє працювати із документацією таким чином що сторінки та розділи можна копіювати із усім їх наповненням, тому використання шаблонів спрощується в декілька разів, що надає навіть більшої цінності використанню такого прийому. Використання єдиного програмного забезпечення та шаблонів для моделювання діаграм також є необхідними змінами. Так само як і у випадку із текстовими форматами представлення, шаблони діаграм забезпечать однорідність та однозначність оформлення. Такий підхід відповідає методу Скрам оскільки економить час витрачений на розробку власних шаблонів та вивчення написаної документації.

Хоча для використання прийому **визначення джерел вимог** в документації наявні відповідні поля для заповнення цей прийом не використовується в компанії. Це пов'язано з тим що вимоги надходять виключно від замовника, а тому потреби у визначенні джерел немає, проте якщо в майбутньому організація планує створювати фокус-групи та співпрацювати із представниками користувачів то впровадження цього прийому є обов'язковим. Це надасть можливість відслідковувати від кого поступила вимога і до кого звертатись у разі її зміни.

Призначення кожній вимозі унікального ідентифікатора (Див. Додаток Б) є ще однією важливою практикою що дозволяє полегшити організацію та ознайомлення із вимогами. Як було вказано у п. 2.3 вже наявна документація по проєкту не мала структури що використовувала унікальні ідентифікатори для кожної вимоги. Рекомендується впровадити такі ідентифікатори якнайшвидше для можливості легкої орієнтації по документації, відслідковування зв'язків між вимогами та розподіл вимог атомарним чином. В інструменті що використовується для ведення документації вимоги варто атомарно розподіляти по сторінкам та призначати унікальний ідентифікатор, склад якого може розробити команда аналітиків. Це надасть можливість посилатись на окрему вимогу за ідентифікатором, а також перевикористовувати вимоги в інших артефактах, такі як сценарії використання, бізнес-правила та інші. Окрім того варто призначати такі ідентифікатори кожному артефакту документації задля звернення до окремого елементу в зручний спосіб.

Прийом документування бізнес-правил хоч і використовується на проєктах організації проте необхідно провести певні зміни у його використанні. Оскільки бізнес-правилами вважаються зовнішні та внутрішні регуляторні документи та правила, а також стандарти індустрії і тп., можна вважати що поняття цього артефакту підміняється в рамках розробки вимог. В цій специфікації під терміном бізнес-правил поєднано одночасно і бізнес-правила, і обмеження що використовуються в системі, що є грубою помилкою та потребує розмежування. Тому необхідними рішеннями є розподіл реальних бізнес-правил від простих обмежень в системі, та опис джерела бізнес-правил в специфікації. Це вирішить проблему підміни понять та надасть можливість вивести бізнес-правила на рівень організації з рівня проєкту. Додатково це забезпечить проведення відповідності певних бізнес-правил до їх офіційних джерел, що буде надійним методом посилання на офіційні регуляторні документи, стандарти та інше.

Визначення нефункціональних вимог як прийом що використовується в процесі документування вимог є достатньо складним, з точки зору кількісної

оцінки таких вимог. Оскільки такі вимоги поєднують якісний опис характеристик продукту вони не підлягають об'єктивній оцінці до безпосереднього впровадження функціоналу та перевірки роботи системи. На проєкті PlatmixSystem такі вимоги не фіксуються що є мінусом та негативно впливає на процес розробки. Звісно цей прийом необхідно впровадити при розробці вимог в ТОВ «Платмастер». Це забезпечить високу якість кінцевого продукту, зменшить ризики невідповідності системи очікуванням користувачів і замовників, а також підвищить задоволеність клієнтів. Впровадження процесу визначення та документування нефункціональних вимог дозволить чітко сформулювати очікування щодо продуктивності, безпеки, надійності, масштабованості та інших критичних характеристик системи, які не є безпосередньо пов'язаними з її функціональними можливостями.

Це також забезпечить краще управління вимогами протягом усього життєвого циклу проєкту, оскільки нефункціональні вимоги часто мають великий вплив на архітектурні рішення та технології, що використовуються при розробці. Наприклад, чітко визначені вимоги до продуктивності можуть вимагати оптимізації коду або вибору певної інфраструктури, що забезпечить необхідний рівень швидкодії.

Четвертою і заключною складовою частиною розробки вимог є **узгодження вимог**. В п. 1.2 було зазначено що цей етап забезпечує перевірку відповідності вимог до поставлених бізнес-цілей. В організації ТОВ «Платмастер» цей етап розробки вимог майже повністю пропускається, що створює значні ризики виробничих простоїв та непорозумінь із замовником. Тому обов'язковим є впровадження цього етапу для перевірки створеної документації як членами команди та замовником, так і іншими зацікавленими сторонами.

Основним рекомендованим прийомом цієї складової є **дослідження документів з вимогами**. Задля забезпечення повноти та якості написаної документації необхідно проводити інспекцію вимог перед їх впровадженням на проєкті. Зацікавлених сторін, замовника та членів проєктної команди необхідно навчити як ефективно перевіряти вимоги. Також потрібно зарегламентувати процес

перевірки що може бути організований як зустріч із відповідною темою що займатиме до 2х годин часу із перервою. Таким чином кожен із споживачів цієї документації зможе ознайомитись із нею та надати відгук або вирішити питання що виникли в процесі перевірки. Узгоджені зміни оперативно будуть внесені у документацію. Аналітик зможе зробити висновки та змінити або покращити процес документації в майбутньому. Наразі лише аналітик вимог проводить перевірку вимог перед їх впровадженням, що видно на діаграмі рис. 2.6. У зв'язку з цим вимоги хоч і проходять перевірку але об'єктивно її не можна назвати, саме тому варто впровадити повноцінну перевірку вимог зацікавленими сторонами та проєктною командою після їх документування. Впровадження етапу узгодження вимог забезпечить відповідність описаної документації баченню та побажанням клієнта, а також надасть можливість команді поєднано ознайомитись із вимогами та надати коментарі щодо поліпшення формату написання.

Тестування вимог є прийомом, що полягає в створенні тестів для перевірки описаних вимог та правильності реалізації функціоналу. Іншими словами аналітик вимог описує окремі тести що відображають очікувану поведінку систему в різноманітних умовах. Через відсутність такої практики в компанії та через недостачу людських ресурсів, процедуру тестування вимог пропонується впровадити пізніше після розширення штабу працівників. В майбутньому необхідно регламентувати процедуру та шаблони для написання тестів до вимог, та запровадити відповідний формат зустрічі для проходження тестів разом із замовником. Також орієнтуючись на принципи методу Скрам формат написання тестів може бути спрощений до необхідного рівня деталізації заради зменшення витрат часу.

Прийом визначення критеріїв приймання у зв'язці із попереднім прийомом тестування вимог надає можливість замовнику бути ще більше залученим до тестування продукту, це забезпечує своєчасну перевірку відповідності продукту його баченню. В організації ТОВ «Платмастер» цей прийом так само як і попередній варто впроваджувати після розширення проєктної

команди. Проте дану практику можна впровадити і зараз з орієнтацією на швидку доставку результатів. Під час зустрічей з перевірки вимог, аналітик додатково може виявити разом із замовником за якими критеріями другий буде оцінювати прийнятність розробленого функціоналу, це можуть бути описані сценарії використання, узгоджені прототипи дизайну та інше. Після завершення спринта та доставки результатів за методом Скрам, аналітик вимог та замовник можуть за описаними критеріями приймання оцінити відповідність розробленого функціоналу та одразу виявити зміни які необхідно внести, додаткові вимоги, помилки або невідповідність нефункціональним вимогам.

Останній рекомендований прийом - **моделювання вимог**. Він полягає у використанні спеціалізованих програмних засобів для попереднього моделювання системи на основі зібраних вимог. Даний прийом здебільшого застосовується під час написання вимог до нового продукту розробка якого ще не почалась, тому у випадку компанії ТОВ «Платмастер» наразі немає необхідності його застосовувати. Окрім цього зараз компанія не має аналітиків що мають досвід у моделюванні систем за допомогою спеціалізованих програм, а це означає що до того як таке програмне забезпечення почне використовуватись персонал повинен буде пройти навчальний курс для опанування такого інструменту.

Застосувавши запропоновані вдосконалення, графічні представлення діяльності та процесів «як буде» виглядатимуть як зображено на рис. 3.1 та рис. 3.2.

На рис. 3.1 видно що в потоці з'явилися додаткові діяльності та умови. Після документування вимог додалась діяльність їх перевірки, що, як зазначалось раніше, забезпечить узгодженість вимог до бачення та побажань замовника і інших зацікавлених сторін, що зробить процес розробки більш прозорим, а також забезпечить відповідність вимог.

користувачами наразі недоступні для впровадження через обмежені ресурси організації, саме тому рекомендується впровадження таких прийомів у майбутньому. На етапі аналізу вимог прийоми що використовуються також потребують певних змін задля відповідності принципам Скрам, таким чином деякі з них навіть можна опустити для поточних стадій розробки продукту. Рекомендовані прийоми етапу документування варто впровадити в повному обсязі, це повинно укріпити та стандартизувати інфраструктуру документації, оскільки інструмент ведення документації дозволяє не тільки організувати використання усіх цих прийомів, але й оптимізувати їх. Етап перевірки вимог як і зазначалось необхідно впровадити хоча б з використанням прийому рецензії вимог, це надасть замовнику можливість перевірити правильність вимог перед їх реалізацією. Загалом компанії слід розглянути впровадження деяких прийомів прямо зараз, це підніме якість написаних вимог та розуміння усіх зацікавлених сторін щодо суті роботи аналітика.

3.2 Систематизація та структуризація процесу управління вимогами

Управління вимогами є процесом, що супроводжує весь життєвий цикл вимог і включає аналіз, внесення змін та підготовку до їх впровадження. У п. 2.3 було зазначено що наявність регламентованого процесу управління вимогами зменшує негативний вплив змін і забезпечує їх своєчасну та організовану інтеграцію. Саме через це першочерговим та обов'язковим є завдання регламентації цього процесу, задля усунення плутанини та простоїв через невизначеність дій.

Першим прийомом що використовується в управлінні вимогами є **визначення процесу управління змінами**. Як зазначалось раніше такий процес не був регламентований в ТОВ «Платмастер», тому аналітику вимог у співпраці зі своїми колегами по відділу необхідно узгодити алгоритм дій та набір прийомів що будуть використовуватись при управлінні вимогами. Потік бізнес-процесу управління вимогами що зображений на діаграмі рис. 2.8 описує ситуацію «як є», тому внесення змін до нього є необхідним завданням. Як і в ситуації із процесом розробки вимог пропонується додати етап перевірки вимог замовником та зацікавленими сторонами після документування змін, це повинно надати можливість остаточно узгодити зміни перед їх впровадженням в системі. Додатково необхідно зробити завдання аналізу та обговорення змін із проєктною командою обов'язковим кроком при внесенні змін, таким чином оцінка здійсненності та витрат ресурсів з боку команди розробників може відкоригувати зміни які планується внести.

Після узгодження регламенту процесу управління вимогами аналітик, як і у випадку із процесом розробки, має презентувати відповідний документ замовнику та зацікавленим сторонам для його узгодження із ними.

Визначення базової версії наборів вимог та управління версіями вимог, відстеження хронології змін та відстеження стану вимог, є наступними прийомами управління вимогами. Організація має ряд проблем які можна вирішити за допомогою цих практик. В п. 2.3 було описано проблеми що значним чином впливають на управління вимогами, серед них були проблеми неоднорідності формату запису змін, відсутність версіювання вимог та створення шаблонів документації на льоту.

У першому випадку проблема полягає в тому що через невизначеність щодо формату запису змін до документації інколи зміни записувались поверх існуючої документації і виокремлювались за допомогою інструментів редагування тексту, що призводить до нагромадження інформації на одній сторінці та ускладнення сприйняття інформації при читанні. Інколи зміни записувались на окремих

сторінках із довільним форматом запису, що призводить до порушення структури документації та ускладнює відслідковування стану вимог та їх впровадження. Вирішенням цієї проблеми може бути використання одного формату запису змін таким чином – необхідно створити центральний вузол, тобто сторінку на яку можна додати таблицю для фіксації усіх змін, дати їх внесення, їх стану та посилання на відповідний розділ або сторінку. Такий центральний вузол гратиме роль організаційної ланки. Окрім цього при безпосередньому внесенні змін варто створювати копії сторінок, вносити зміни та зберігати такі копії як нові версії сторінок, тобто в результаті використання такого підходу в документації зберігатимуться і базові версії вимог і їх нові версії із внесеними змінами. Таким чином створення нових версій документів з внесеними змінами забезпечує збереження історії змін і полегшує їх відстеження.

У другому випадку під час запису змін не використовувались стандартизовані шаблони документації, що призвело до неоднорідності документації та ускладнило процес ознайомлення для її споживачів. Якщо впровадити практики що описані в попередньому абзаці ця проблема усунеться сама собою, оскільки в кожній версії документації буде використаний єдиний формат сторінок, а для центрального організаційного вузлу буде використовуватись єдиний шаблон.

Прийом створення матриці зв'язків вимог певним чином схожий на рішення створення центрального вузлу в документації, проте таке рішення націлене не на окремі інкременти змін що будуть фіксуватись, а на усі вимоги загалом. Його впровадження є вкрай необхідним особливо через специфіку та складність системи що розробляється зараз. Варто створити шаблон такої матриці та розмістити в окремому розділі документації. Надалі необхідно його заповнити існуючими вимогами та видами зв'язків між ними, наприклад – «впливає на», «включено до», «блокує», «ідея для» та інші. Після впровадження такої практики аналітику та проєктному менеджеру буде зручно управляти вимогами та задачами

по проєкту, а інші зацікавлені сторони та замовник завжди зможуть отримати актуальну інформацію щодо стану певних вимог.

Варто розглянути яким чином запропоновані покращення вплинуть на графічні представлення процесу управління вимогами. Вони наведені нижче – рис. 3.3 та рис. 3.4.

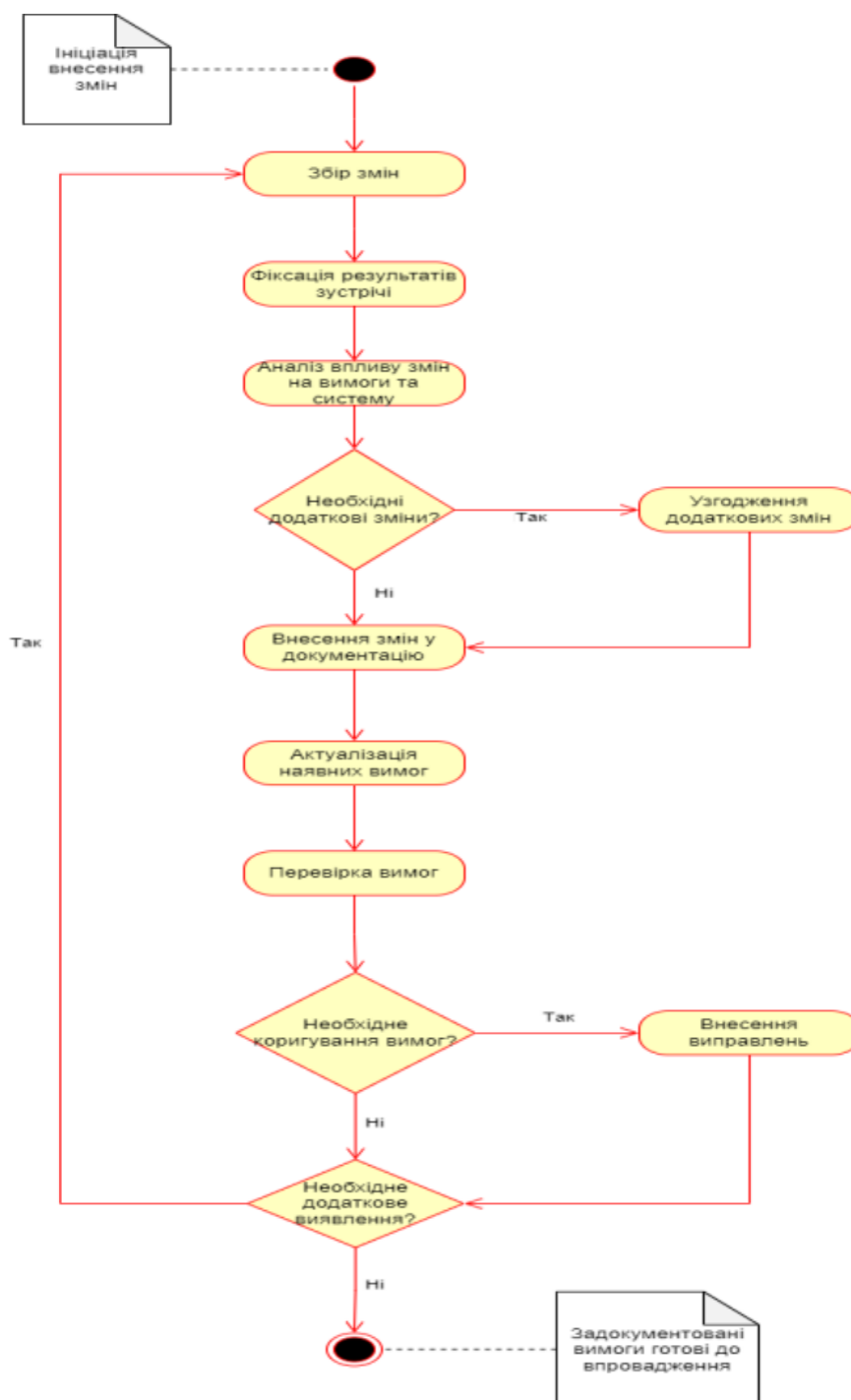


Рисунок 3.3 – Діаграма діяльності – Управління вимогами «як буде»

Джерело: розроблено автором

3.3 Інтеграція процесів розробки та управління вимогами в систему проєктного управління

Оптимізація залежностей між процесами розробки, управління вимогами та проєктним управлінням є критично важливою для ефективної роботи проєктної команди над продуктом. Усі ці процеси тісно пов'язані між собою, а їх неузгодженість може призвести до простоїв, збільшення витрат, низької якості продукту та невдоволення замовника. Системний підхід до оптимізації дозволяє створити чітку структуру, яка забезпечує прозорість, ефективну комунікацію, знижує ризики та підвищує продуктивність команди.

В контексті використання методу Скрам, важливою є інтеграція управління вимогами та розробки у єдиний процес, який відповідає гнучкому підходу. Скрам пропонує структуру, яка складається з коротких ітерацій (спринтів), регулярних зустрічей для планування розробки, огляду реалізовано функціоналу та постійного вдосконалення. Цей метод забезпечує адаптивність і дозволяє швидко реагувати на зміни вимог, що особливо важливо на проєктах гнучкої розробки.

На проєкті PlatmixSystem застосований гнучкий підхід до розробки за методом Скрам, детальний опис прийомів за цим методом що використовуються на проєкті описано в п. 2.2, там зазначено що проєктна команда проводить короткі щоденні зустрічі на початку робочого дня для обговорення виконаних та поточних задач – стендап. Наразі аналітик вимог не залучений до цих зустрічей через що він не знає актуального стану справ по розробці та має окремо дізнаватись про це в проєктного менеджера. Пропонується залучити аналітика до цих зустрічей для того щоб він розумів над якими задачами працюють розробники та тестувальник, а також міг сам відзвітувати якими задачами він займається. Це також допоможе налагодити регулярний контакт із проєктною командою для обговорення питань та проблем що могли виникнути в процесі роботи.

Необхідно залучити аналітика і до демонстрації функціоналу що проводиться в кінці спринта. Аналітик при підготовці до демонстрації може допомогти визначити на чому саме варто робити акцент, а які моменти потребують доробок. Безпосередньо під час презентації аналітик зможе відповісти на питання щодо функціоналу які можуть виникнути у замовника, а проєктний менеджер розповість про план на майбутній спринт. Окрім цього під час проведення зустрічі замовник одразу може висунути вимоги до змін або просто нові вимог по функціоналу, які аналітик може зафіксувати для подальшого виявлення та обговорення на окремій зустрічі.

Додатково варто піти далі та пов'язати процес управління проєктом із задачами що виконує аналітик, тобто пропонується створити для відділу аналітики окремий простір в застосунку Jira де співробітники зможуть заводити собі задачі, оцінювати час на їх виконання та вести їх по модифікованій Скрам дошці (Див. Додаток В) що буде схожа на дошку яку використовують розробники, із стовпцями: «TO DO» (до виконання), «IN PROGRESS» (в процесі), «FOR APPROVAL» (для узгодження), «DONE» (готово). За допомогою впровадження такого прийому проєктного управління в процес роботи аналітиків, проєктний менеджер зможе цілісно слідкувати та планувати процеси розробки продукту та вимог, оскільки невизначеність щодо витрат часу на опис аналітики буде виключена.

Співпраця проєктного менеджера та аналітика є дуже важливим компонентом для успішного управління проєктом. Проєктному менеджеру та аналітику вимог необхідно почати тісно співпрацювати задля оптимізації процесів розробки та управління продукту та вимог, оскільки аналітик краще розуміє концепцію та функціонал продукту, а проєктний менеджер краще орієнтується в часі та ресурсах, що витрачаються на розробку. Така співпраця дозволяє забезпечити, те що вимоги будуть чітко визначені та зрозумілі всім учасникам команди, а також, що процеси розробки будуть прозорі та ефективно сплановані.

Аналітик, завдяки своєму глибокому розумінню функціональних та нефункціональних вимог, може надати цінну інформацію про критичні аспекти

продукту для користувачів та про зміни, що можуть найбільше вплинути на кінцевий результат. У свою чергу, проєктний менеджер, володіючи знаннями про часові рамки, бюджет та обмеження ресурсів, може допомогти визначити пріоритетність вимог та оптимально розподілити ресурси для досягнення цілей проєкту.

Співпраця між цими двома ролями сприяє створенню більш реалістичних планів, швидшому виявленню та вирішенню потенційних проблем, а також забезпечує узгодженість дій всієї проєктної команди. Це підвищує якість продукту, зменшує ризики та покращує загальну ефективність управління проєктом.

ВИСНОВКИ

У процесі написання роботи було детально проаналізовано та досліджено різні аспекти розробки та управління вимогами до програмних продуктів, зокрема в контексті використання гнучкого методу розробки Скрам. Була розглянута роль та суть вимог до програмного продукту в процесі його розробки.

Основна увага приділялася рекомендованим прийомам що застосовуються в процесі роботи з вимогами, принципам системного підходу та їх практичному застосуванню для оптимізації залежностей між процесами розробки, управління вимогами та проєктним управлінням.

Системний підхід до розробки та управління вимогами, з його принципами цілісності, взаємозв'язку елементів, ієрархічності, невизначеності та динамічності, забезпечує комплексний та гнучкий підхід до роботи з вимогами на всіх етапах життєвого циклу проєкту. Цей підхід дозволяє не тільки враховувати взаємодію між окремими вимогами та компонентами системи, але й адаптуватися до змін і забезпечувати ефективну координацію між різними учасниками проєкту.

Була проаналізована діяльність компанії ТОВ «Платмастер» та процеси розробки та управління вимогами, а також процес проєктного управління на прикладі проєкту PlatmixSystem. Був розглянутий програмний інструментарій що включає додатки – Jira, Confluence, Slack та інші. Було досліджено недоліки та проблеми в виробничих процесах та процесах документування вимог.

В роботі було запропоновано конкретні кроки для впровадження покращень в процес розробки вимог на основі методології Скрам. Було виявлено низку проблем, зокрема відсутність фіксації нефункціональних вимог, відсутність узгоджених шаблонів для документації та відсутність використання прийомів співпраці із користувачами. Впровадження регулярних перевірок вимог, стандартизація шаблонів документації, співпраця із різними класами користувачів, використання спеціалізованих інструментів для управління вимогами, таких як

Confluence та Jira сприяють покращенню прозорості, узгодженості та ефективності роботи проєктної команди.

Також важливим аспектом було дослідження та вдосконалення процесу управління вимогами в ТОВ «Платмастер», де неузгоджений процес управління вимогами та неструктурований підхід до фіксації змін стали основними проблемами. Запропоновані покращення, такі як впровадження регламентованого процесу управління вимогами, використання сучасних інструментів для моніторингу змін, а також створення матриці зв'язків вимог дозволяють підвищити якість управління вимогами та зменшити негативний вплив змін на проєкт.

Таким чином, результати дослідження підтвердили важливість застосування системного підходу до управління вимогами для успішної розробки програмних продуктів. Впровадження запропонованих методів і підходів дозволяє досягти більшої узгодженості, ефективності та якості в процесах розробки та управління вимогами, і як результат в процесах проєктного управління та розробки програмного продукту загалом, що, в свою чергу, сприяє задоволенню потреб замовника та досягненню стратегічних цілей організації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sommerville, I., Sawyer, P. Requirements Engineering: A Good Practice Guide. Chichester: Wiley, 1997. 416 p.
2. Wiegers, K., Beatty, J. Software Requirements. Developer Best Practices. 3rd ed. Redmond: Microsoft Press, 2013. 672 p.
3. ISO/IEC/IEEE 29148:2018(en) Systems and software engineering — Life cycle processes — Requirements engineering // ISO Online Browsing Platform. URL: <https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:29148:ed-2:v1:en> (дата звернення: 17.03.2024)
4. Gottesdiener E. Requirements by Collaboration: Workshops for Defining Needs. Boston: Addison-Wesley, 2002. 304 p.
5. Методика та організація наукових досліджень. Системний підхід // Lib Net. URL: http://lib-net.com/content/11015_Sistemnyi_pidhid.html (дата звернення: 24.03.2024).
6. Системний підхід // Вікіпедія. URL: https://uk.wikipedia.org/wiki/Системний_підхід (дата звернення: 25.03.2024).
7. Системний підхід як метод наукового пізнання світу. Реферат // Osvita.ua. URL: <https://osvita.ua/vnz/reports/philosophy/12876/> (дата звернення: 25.03.2024).
8. ПЛАТМАСТЕР // YouControl — сервіс перевірки контрагентів. URL: https://youcontrol.com.ua/catalog/company_details/42624957/ (дата звернення: 25.04.2024).
9. 10 Типів організаційних структур компаній // Worksection // Блог. URL: <https://worksection.com/ua/blog/10-types-of-organizational-structures-for-companies.html> (дата звернення: 25.04.2024).

- 10.Проектна організаційна структура управління // Моя Освіта. URL:
https://moyaosvita.com.ua/menedzhment/proektna-organizacijna-struktura-upravlinnya/#google_vignette (дата звернення: 25.04.2024).
- 11.What is scrum? // Atlassian. URL: <https://www.atlassian.com/agile/scrum> (дата звернення: 25.04.2024).
- 12.What is a scrum master & their responsibilities? // Atlassian. URL:
<https://www.atlassian.com/agile/scrum/scrum-master> (дата звернення: 25.04.2024).
- 13.Agile scrum roles and responsibilities // The product owner: Setting clear direction // Atlassian. URL: <https://www.atlassian.com/agile/scrum/roles> (дата звернення: 25.04.2024).
14. The development team: Redefining “developer” // Agile scrum roles and responsibilities // Atlassian. URL: <https://www.atlassian.com/agile/scrum/roles> (дата звернення: 25.04.2024).
- 15.What is a stand up meeting & tips to run one // Atlassian. URL:
<https://www.atlassian.com/agile/scrum/standups> (дата звернення: 25.04.2024).
- 16.Slack // Wikipedia. URL: <https://uk.wikipedia.org/wiki/Slack> (дата звернення: 25.04.2024).
- 17.Jira // Wikipedia. URL: <https://uk.wikipedia.org/wiki/Jira> (дата звернення: 25.04.2024).
- 18.Jira boards // Jira // Atlassian. URL:
<https://www.atlassian.com/software/jira/features/scrum-boards> (дата звернення: 25.04.2024).
- 19.Three ingredients for great software releases // Atlassian. URL:
<https://www.atlassian.com/agile/software-development/release> (дата звернення: 25.04.2024).
- 20.Scrum sprints // Atlassian. URL: <https://www.atlassian.com/agile/scrum/sprints> (дата звернення: 25.04.2024).

21. Confluence (software) // Wikipedia. URL:
[https://en.wikipedia.org/wiki/Confluence_\(software\)](https://en.wikipedia.org/wiki/Confluence_(software)) (дата звернення: 25.04.2024).
22. UML Activity Diagram Tutorial // Lucidchart. URL:
<https://www.lucidchart.com/pages/uml-activity-diagram> (дата звернення: 25.04.2024).
23. BPMN & BPMN 2.0 Tutorial // Lucidchart. URL:
<https://www.lucidchart.com/pages/bpmn-bpmn-2-0-tutorial> (дата звернення: 25.04.2024).
24. What is BABOK (Business Analysis Body of Knowledge)? A Comprehensive Guide // Modern Requirements. URL:
<https://www.modernrequirements.com/blogs/babok-business-analysis-body-of-knowledge/> (дата звернення: 25.04.2024).
25. Не BABOK єдиним – огляд інших Body of Knowledge (BoK) для бізнес-аналітиків // Про Бізнес-Аналіз Українською Стартова сторінка українськомовних бізнес-аналітиків. URL: <https://www.ba.in.ua/2023/09/25/ne-babok-yedynym-oglyad-inshyh-body-of-knowledge-bok-dlya-biznes-analitykiv/> (дата звернення: 25.04.2024).
26. International Institute of Business Analysis. A Guide to the Business Analysis Body of Knowledge (BABOK Guide). 3rd ed. International Institute of Business Analysis, 2015. 512 p.
27. Що таке управління вимогами: визначення, переваги, найкращі інструменти // Visure. URL: <https://visuresolutions.com/uk/блог/управління-вимогами/> (дата звернення: 25.04.2024).
28. 01м - Моделі і методи проектування інформаційних систем // Sumdu. URL: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20160217112601/165292/index.html#p13 (дата звернення: 25.04.2024).

29. Грицюк Ю. І., Нємова О. А. Особливості формулювання вимог до програмного забезпечення. Науковий вісник НЛТУ України. 2018, т. 28, № 7. С. 135–148.
30. Хто такий QA та яка його роль у процесі розробки // Web Academy Media. URL: <https://web-academy.ua/blog/junior/role-of-qa-in-software-development> (дата звернення: 25.04.2024).
31. Функціональні та нефункціональні вимоги // Visure. URL: <https://visuresolutions.com/uk/посібник-з-відстеження-управління-вимогами/функціональні-та-нефункціональні-вимоги/> (дата звернення: 25.04.2024).
32. Стадії циклу розробки ПЗ // QALight Центр підготовки ІТ фахівців. URL: <https://qalight.ua/baza-znaniy/stadiyi-tsiklu-rozrobki-pz/> (дата звернення: 25.04.2024).
33. Цибульник С. О., Барандич К. С. Технології розроблення програмного забезпечення: підручник. Київ: КПІ ім. Ігоря Сікорського, 2022. 270 с.
34. Популярні життєві цикли розробки ПЗ // QATestLab training center. URL: <https://training.qatestlab.com/blog/technical-articles/popular-software-development-life-cycles/> (дата звернення: 25.04.2024).
35. Люшенко Л. А., Хіцко Я. В. Розробка та аналіз вимог до програмного забезпечення: навч. посіб. Київ: КПІ ім. Ігоря Сікорського, 2020. 64 с.
36. Що таке специфікація вимог: визначення, найкращі інструменти та методи | Гід // Visure. URL: <https://visuresolutions.com/uk/блог/специфікація-вимог/> (дата звернення: 25.04.2024).
37. Менеджмент у сфері програмного забезпечення // Вікіпедія. URL: https://uk.wikipedia.org/wiki/Менеджмент_у_сфері_програмного_забезпечення (дата звернення: 25.04.2024).
38. Критерії якості вимог до ІТ-продукту: що це, які існують і навіщо потрібні? // IAMPM. URL: <https://iampm.club/ua/blog/kriteriyi-yakosti->

- [vimog-do-it-produktu-shho-cze-yaki-isnuyut-i-navishho-potribni/](#) (дата звернення: 25.04.2024).
39. Вимоги до програмного забезпечення // Вікіпедія. URL: [https://uk.wikipedia.org/wiki/Вимоги_до_програмного_забезпечення](#) (дата звернення: 25.04.2024).
40. Мазак А. В. Методологія системного підходу та наукових досліджень: Конспект лекцій для слухачів магістратури за спеціальністю "Державна служба". Івано-Франківськ, 2009. 193 с.
41. Основні поняття та категорії управління якістю // Конспект Лекцій. URL: [https://elib.tsatu.edu.ua/dep/ate/tpzpsg_2/page2.html](#) (дата звернення: 25.04.2024).
42. Кустовська О. В. Методологія системного підходу та наукових досліджень: курс лекцій. Тернопіль: Економічна думка, 2005. 124 с.
43. Жигалкевич Ж. М., Чухліб В. Є. Управління проектами та їх ризиками: підходи та методи. *Приазовський економічний вісник*. 2019. Вип. 6. С. 126-130.
44. Навчальний посібник з дисципліни «Технології розробки програмного забезпечення» для студентів спеціальності 123 «Комп'ютерна інженерія». Полтава: ПолтНТУ, 2017. 218 с.
45. Лучко Г. Й., Лебідь Т. В., Когут І. В. Сучасний стан та проблеми розвитку проектного менеджменту в Україні. *Мукачевський державний університет. Економіка і суспільство*. 2017. Вип. 9. С. 501-507.
46. Андрєєва Т. Є., Петровська Т. Е., Титар Т. С. Проектний менеджмент як засіб досягнення мети підприємства. *Вісник економіки транспорту і промисловості*. 2011. Вип. 34. С. 364-370.
47. Розробка програмного забезпечення // Дія.Бізнес. URL: [https://business.diia.gov.ua/idea/it-ta-telekom/rozrobka-programnogo-zabezpecenna](#) (дата звернення: 25.04.2024).

- 48.Проектний менеджмент (управління проектами) // переріслянська територіальна громада. URL: <https://pererislyanska-gromada.gov.ua/proektnij-menedzhment-upravlinnya-proektami-1532185544/> (дата звернення: 25.04.2024).
- 49.Project Management: 10 базових книг з трьох ключових тем // EPAM Campus. URL: <https://training.epam.ua/ua/blog/571> (дата звернення: 25.04.2024).
- 50.Business Process Modeling Guide // MEGA. URL: <https://www.mega.com/blog/business-process-modeling-guide> (дата звернення: 25.04.2024).

Додаток А

Шаблони діаграм

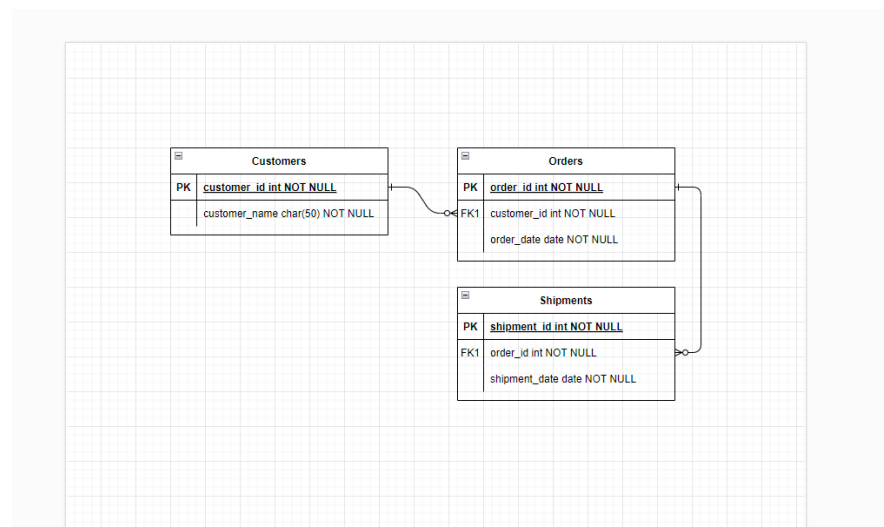


Рисунок А1 – Діаграма сутність зв'язок

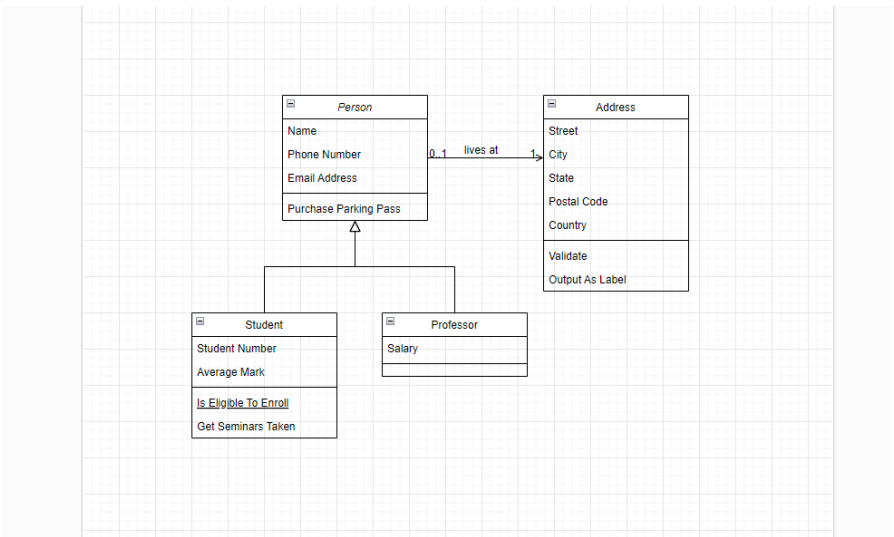


Рисунок А2 – Діаграма класів

Продовження додатку А

Шаблони діаграм

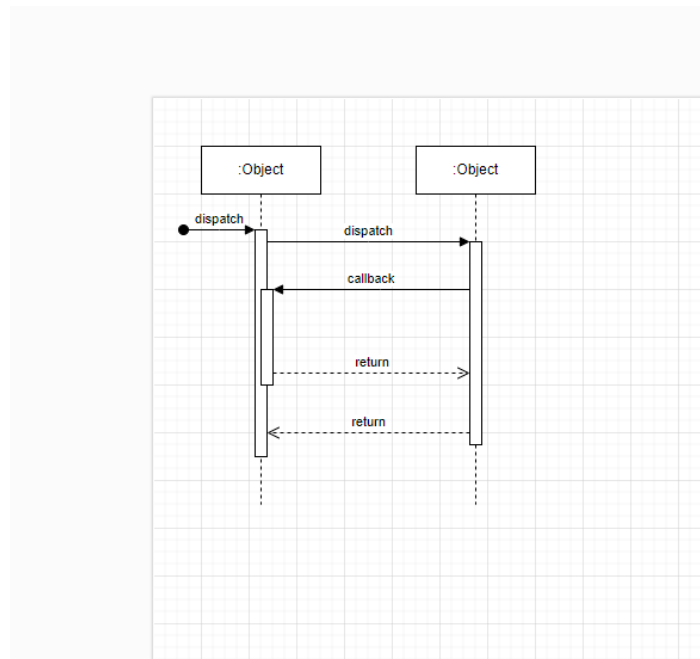


Рисунок А3 – Діаграма послідовності

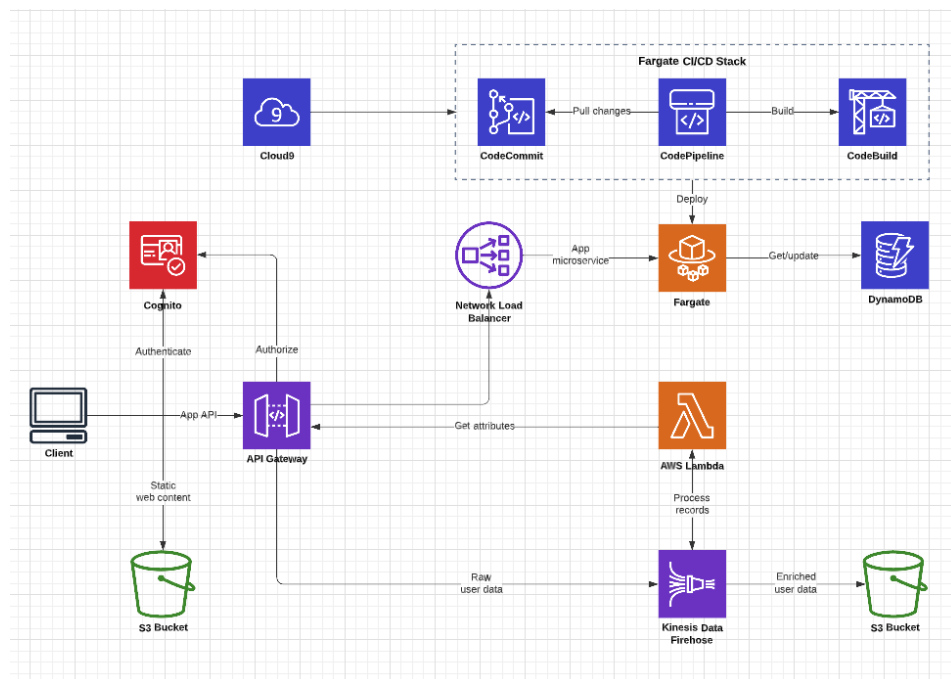


Рисунок А4 – Діаграма архітектури системи

Приклад використання унікального ідентифікатора

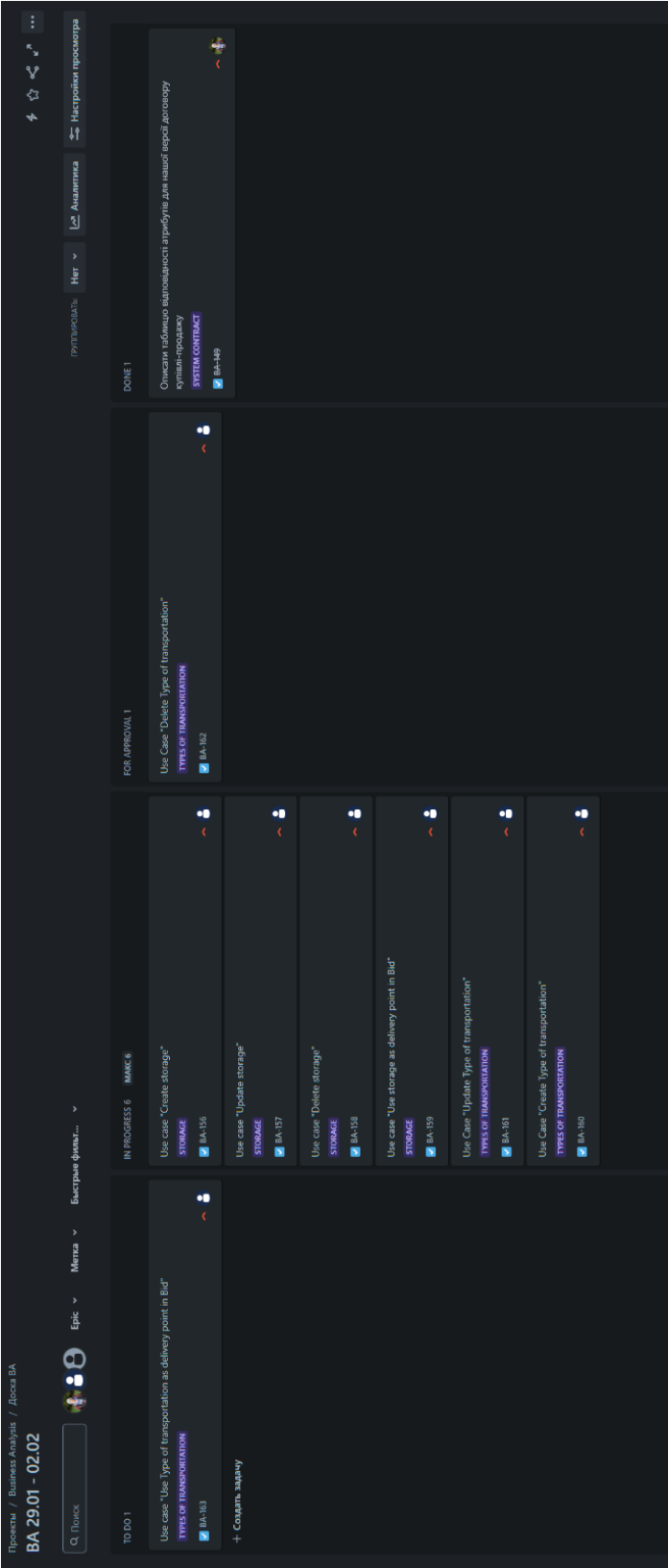
GSFP. EDM. BC. GE. R. UR. EDM-GE-URReq-4

Владелец: Ярослав Скарженский
Последнее обновление: апр. 02, 2024 • Просмотреть 1 пользователя

EReq ID	Req Description	Diag Step Name	Status	Req Type	Relates User Use Case		Relates System Use Case		Related User Interface Spec		Related UI Design Wireframes	
					Cases ID	Status	Cases ID	Status	Spec ID	Status	Wireframe Link	Status
EDM-GE-URReq-4	Я як Користувач повинен мати можливість завантажувати протокол торгів по моїй Смарт Зав'язці	"Download Trade Protocol"	"designed"	Online	EDM-GE-UC-4	"designed"						

Додаток В

Скрам дошка



Ім'я користувача:
Комп'ютерної математики та інформаційної безпеки...

ID перевірки:
1016281676

Дата перевірки:
25.05.2024 10:28:47 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
25.05.2024 10:30:11 EEST

ID користувача:
100005746

Назва документа: Диплом_Скаржевський

Кількість сторінок: 84 Кількість слів: 18681 Кількість символів: 140935 Розмір файлу: 1.35 MB ID файлу: 1016074480

1.76% Схожість

Найбільша схожість: 0.65% з Інтернет-джерелом (https://www.fg.gov.ua/storage/files/6-17-ppa_1631541468.xls)

1.33% Джерела з Інтернету 21

Сторінка 86

1.63% Джерела з Бібліотеки 298

Сторінка 86

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0% Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи

1