Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: **Розробка методів і системи управління зрілістю вимог на етапах життєвого циклу програмного забезпечення із застосуванням мови програмування PHP**

Виконав: студент (ка) 6 курсу, групи СПм-61
спеціальності 121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

	(підпис)	Василишин В.І. (прізвище та ініціали)
Керівник	(підпис)	Пастух О.А. (прізвище та ініціали)
Нормоконтроль	(підпис)	Стоянов Ю.М. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Петрик М.Р. (прізвище та ініціали)
Рецензент	(підпис)	(прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

(повне найменування вищого навчального закладу)

Факультет комп'ютерно-інформаційних систем і програмної інженерії

(назва факультету)

Кафедра програмної інженерії

(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту (роботи)

магістра

(освітньо-кваліфікаційний рівень)

на тему: Методи і система управління зрілістю вимог на етапах життєвого циклу програмного забезпечення із застосуванням мови програмування PHP

Виконав: студент 6 курсу, групи СПм-61

Спеціальність

121 «Програмна інженерія»

(шифр і назва напряму підготовки, спеціальності)

		Василишин В.І.
	(підпис)	(прізвище та ініціали)
Керівник		Пастух О.А.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Стоянов Ю.М.
	(підпис)	(прізвище та ініціали)
Рецензент		
	(підпис)	(прізвище та ініціали)

м. Тернопіль – 2021

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра _____Програмної інженерії _____

Рівень вищої освіти - другий (магістерський)

Спеціальність _121-Інженерія програмного забезпечення_____
(код і повна назва)

Тип програми освітньо-наукова програма

Освітня програма_Інженерія програмного забезпечення_

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 20 ___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові _*Василишину Володимиру Івановичу*_
(прізвище, ім'я, по батькові)

1. Тема роботи _*Методи і система управління зрілістю вимог на етапах життєвого циклу програмного забезпечення із застосуванням мови програмування PHP*_____

_____

затверджена наказом університету від "___" _____2021 р № _____

2. Термін подання студентом роботи до екзаменаційної комісії

___ грудня 2021 р.

3. Вихідні дані до роботи *моделі представлення зрілості вимог, технологія і середовище об'єктно-орієнтованого програмування та проектування, рекомендації стандартів якості*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз предметної області і постановка задачі, аналіз моделей зрілості вимог, метод управління зрілістю вимог на етапах життєвого циклу, архітектура*

програмного засобу зрілістю вимогами, реалізація системи управління зрілістю вимог, апробація запропонованого методу і розробленого засобу

5 Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Охорона праці та	Осухівська Г.М.		
безпека в надзвичайних ситуаціях	Клепчик В.М.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1.	Аналіз підходів і методів інженерії вимог	10 жовтня 2021 р.	
2.	Обґрунтування та формалізація моделі управління зрілістю вимог	30 жовтня 2021 р.	
3.	Програмна система збору потреб та управління зрілістю вимог програмного забезпечення	24 листопада 2021 р.	
4.	Охорона праці та безпека в надзвичайних ситуаціях	1 грудня 2021 р.	
5.	Підготовка пояснювальної записки	6 грудня 2021 р.	
6.	Підготовка презентації та доповіді	8 грудня 2021 р.	
7.	Попередній захист	10 грудня 2021 р.	
8.	Нормоконтроль, рецензування	12 грудня 2021 р.	
9.	Занесення диплома в електронний архів	20 грудня 2021 р.	
10	Допуск до захисту у зав. кафедри		

Дата видачі завдання _____ _____2021 р.

Студент _____ *Василишин В.І.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *проф. Пастух О.А.*
(підпис) (прізвище та ініціали)

РЕФЕРАТ/ABSTRACT

Атестаційна робота магістра містить: 105 с., 33 рис., 4 табл., 26 джер.

КЛЮЧОВІ СЛОВА: МОДЕЛЬ, ЗРІЛІСТЬ, УПРАВЛІННЯ, ВИМОГИ, ЖИТТЄВИЙ ЦИКЛ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Метою кваліфікаційної роботи є дослідження методів і створення системи управління зрілістю вимогами програмного забезпечення на етапах життєвого циклу.

У кваліфікаційній роботі проаналізовано основні поняття та класифікацію вимог до програмного забезпечення, визначено особливості процесів інженерії вимог.

Обґрунтовано модель зрілості RMM (Requirements Maturity Model) для управління зрілістю вимог до програмного забезпечення та запропоновано її модифікацію шляхом імплементації рекомендацій стандарту ISO 25010 на вищих рівнях ієрархії моделі.

Засобами мови PHP реалізовано систему управління зрілістю вимог.

KEYWORDS: MODEL, MATURITY, MANAGEMENT, REQUIREMENTS, LIFE CYCLE, SOFTWARE.

In the thesis the basic concepts and classification of requirements to the software are analyzed, features of processes of engineering of requirements are defined.

The RMM maturity model (Requirements Maturity Model) for software maturity management is substantiated and its modification is proposed by implementing the recommendations of the ISO 25010 standard at the highest levels of the model hierarchy.

The requirements maturity management system is implemented using PHP language tools.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД	База даних
ПЗ	Програмне забезпечення
BE	Back End
CASE	Computer Aided Software Engineering
ER	Entity Relations
FE	Front End
UML	Unified Modeling Language

ВСТУП

Актуальність теми. На сьогодні інженерія програмного забезпечення є найбільш технологічною сферою у галузі інформаційних технологій та забезпечує можливість створення потужних багатофункціональних, надійних і продуктивних комплексів, орієнтованих на підтримку бізнес-процесів у різних видах діяльності. Без програмного забезпечення, що виконує функції її «мозку», неможливе функціонування будь-яких сучасних комп'ютерних систем, а рутинні операції практично неможливо автоматизувати. Процес створення програмних комплексів вимагає застосування різних методів і засобів, які спрямовані на забезпечення ефективного використання ресурсів з одночасним виконанням етапів проекту на основі обраної розробником моделі життєвого циклу.

Важливими аспектами реалізації якісного програмного забезпечення є інтеграція у процеси життєвого циклу, визначені стандартом ISO/IEC 12207, додаткових заходів, які орієнтовані на збір метаданих і дозволяють впроваджувати ефективні управлінські рішення. Фундаментальним процесом життєвого циклу не важливо від використовуваної моделі є процес виявлення та аналізу вимог на початку виконання проекту і моніторинг ступеня їх виконання на наступних ітераціях проекту.

Для оцінювання ефективності команди розробників, рівня застосування CASE-засобів, здатності спостереження за вимогами на протязі усього часу виконання проекту доцільно використовувати моделі і методи управління зрілістю. Це дозволить уникати або прогнозувати потенційні ризики та значно підвищити як якість управління проектом, так і якість самого кінцевого продукту.

Серед моделей зрілості, що використовують в інженерії програмного забезпечення варто відмітити модель CMM та похідні від неї, що в переважній більшості стосуються управління персоналом, ризиками та оцінкою ефективності використання ресурсів при виконанні процесів реалізації проекту.

Дослідженню методів і моделей управління зрілістю вимог при проектуванні програмного забезпечення займаються багато українських (О. Летичевський, О. Харченко, П. Андон, К. Лавріщева) і закордонних (I. Sommervill, W. DeLone, B. Boem, M. Holsted,) вчених. Однак у результатах їхніх працях практично відсутній механізм щодо практичної імплементації моделі зрілості вимог, а також не запропоновано комплексного підходу до управління зрілістю вимог з використанням стандартів щодо їх якості. Тому актуальною задачею на сьогодні є побудова моделі, створення методу і програмної імплементації підтримки процесу управління зрілістю вимогами на стадіях життєвого циклу.

Метою кваліфікаційної роботи є дослідження методів і створення системи управління зрілістю вимогами програмного забезпечення на етапах життєвого циклу.

Досягнення мети роботи відбувається шляхом розв’язання поставлених **задач**:

– аналітичний огляд наукових публікацій і стандартів, що використовуються на етапі виявленні та аналізу вимог до програмного забезпечення;

– обґрунтування та математичне представлення моделі управління зрілістю процесів керування вимогами під час життєвого циклу виконання програмного проекту;

– імплементація процедур підвищення якості рівнів зрілості вимог при управлінні вимогами;

– проектування архітектури підсистеми збору та керування вимогами до програмного забезпечення;

– програмна реалізація прототипу системи збору та управління зрілістю вимог до ПЗ.

Методи дослідження. Для досягнення мети використано наступні методи і технології: аналіз та узагальнення – при проведенні аналізу існуючих підходів, моделей, методів і засобів управління зрілістю процесів розробки ПЗ;

математичного моделювання і теорії множин – при формалізації моделі зрілості вимог і методу підвищення якості її рівнів; проектування, програмування і тестування – при реалізації системи управління зрілістю вимог.

Об’єктом дослідження є процеси управління та забезпечення зрілості вимог на етапах життєвого циклу програмного забезпечення.

Предметом дослідження є моделі, методи та інструментальні засоби оптимізації та підтримки процесу управління зрілістю вимог.

Наукова новизна одержаних результатів.

– уперше запропоновано та модифіковано модель RMM шляхом імплементації рекомендацій стандарту ISO 25010 на вищих рівнях моделі зрілості вимог, що дало змогу забезпечити адекватне їх відображення на наступних стадіях життєвого циклу та оптимізувати ресурси на їх реалізацію, а також реалізувати екосистему CASE-засобів для підвищення ефективності автоматизації процесів життєвого циклу ПЗ;

– набули подальшого розвитку архітектурні та програмні рішення щодо управління зрілістю вимог, які відображають процес розробки програмного забезпечення із застосуванням методології Agile та інтегрованого представлення моделей якості, що дали змогу в повній мірі забезпечити автоматизацію управління зрілістю вимог на основі модифікованої моделі RMM.

Практична цінність результатів дослідження. Впровадження методу і програмної системи управління зрілістю вимогами на етапах життєвого циклу дає змогу підвищити ефективність процесів та якість реалізації проекту за рахунок переходу команди розробників на вищий рівень організації процесів та їх автоматизації.

Публікації. Основні результати дослідження апробовано на X міжнародній науково - технічній конференції молодих учених і студентів «Актуальні задачі сучасних технологій» (24-25 листопада 2021 р.) Тернопільського національного технічного університету імені Івана Пулюя та на IX науково-технічній конференції Тернопільського національного технічного університету імені Івана

Пулюя «Інформаційні моделі, системи та технології» (8-9 грудня 2021 року) у вигляді тез конференцій.

1. Пастух О.А., Василишин В.І., Демида Х.М. Аналіз рівнів зрілості вимог при розробці програмного забезпечення. Матеріали X міжнародної науково - технічної конференції молодих учених і студентів «Актуальні задачі сучасних технологій» (24-25 листопада 2021 р.) Тернопільського національного технічного університету імені Івана Пулюя. Тернопіль: ТНТУ. 2021. С. 123.

2. Пастух О.А., Демида Х.М., Василишин В.І. Характеристики якості даних при функціонуванні програмного забезпечення з базами даних і знань. Матеріали X міжнародної науково - технічної конференції молодих учених і студентів «Актуальні задачі сучасних технологій» (24-25 листопада 2021 р.) Тернопільського національного технічного університету імені Івана Пулюя. Тернопіль: ТНТУ. 2021. С. 124.

Структура роботи. Робота складається з пояснювальної записки та графічної частини. Пояснювальна записка складається з вступу, 4 розділів, висновків, списку використаної літератури та додатків. Обсяг роботи: пояснювальна записка – 105 арк. формату А4, графічна частина – 26 слайди.

1 АНАЛІЗ ПІДХОДІВ І МЕТОДІВ ІНЖЕНЕРІЇ ВИМОГ

1.1. Аналіз основних понять та класифікації вимог до програмного забезпечення

Будь-яка сучасна автоматизована система представляє собою реалізацію функції перетворення та опрацювання вхідної інформації з відповідним виведення результату у зручному для користувача вигляді. Функціональність програмних систем визначається вимогами до функцій та умов їх виконання, а також формату та опрацювання даних. В загальному випадку, вимоги до ПЗ є своєрідним контрактом між зацікавленими сторонами, з одного боку – користувачів або стейкхолдери, а з іншого – команда розробки [1].

Щодо означення вимог, то термінологічно загальноприйнятою їх інтерпретацією є: властивості, які повинні бути реалізованими та якими повинна володіти система щодо забезпечення виконання функцій, визначених стороною замовника. До таких функцій можуть належати автоматизація бізнес-процесів, електронного документообігу, управління даними і їх структурами. Все це необхідно при прийнятті ефективних управлінських та системних рішень. Ядро SWEBOK визначає важливі та основні ідеї, концептуальні особливості у сфері інженерії вимог, які продемонстровано на рис. 1.1.

До сучасних і найбільш поширених та популярних засобів, що використовуються при реалізації процесів інженерії вимог є інструментальні засоби мови UML. Діаграми прецедентів (варіантів використання) дають змогу визначати різні аспекти використання ПЗ з точки зору функціональних вимог, які в подальшому можуть бути трансформовані в проектні архітектурні рішення із застосування діаграм класів або пакетів класів.

Процес визначення вимог відноситься до складних та нетривіальних задач у сфері розробки програмного забезпечення і виконується шляхом комунікації зацікавлених сторін. Сторона замовника формулює потреби та необхідність у

реалізації системи щодо автоматизації процесів і задач, які вона повинна розв'язувати. Паралельно або після обговорення з командою розробників, потреби деталізуються та відповідно класифікуються за видами, що в кінцевому результаті приводить до складання специфікації і технічного завдання на проектування системи.

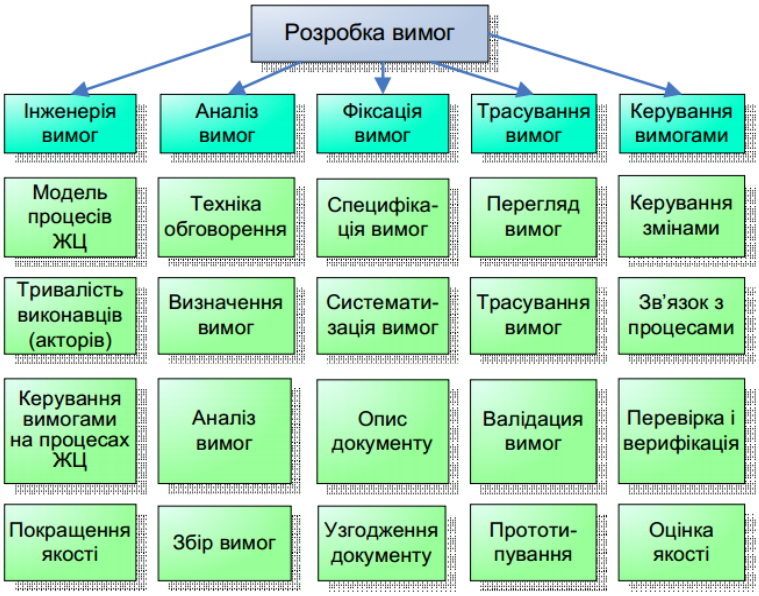


Рисунок 1.1 – Основоположні концепції SWEBOK в інженерії вимог

Характерною особливістю в інженерії вимог є те, що і до сьогодні відсутня усталена термінологія щодо представлення та опису різних типів вимог, зокрема, функціональних та нефункціональних, системних і технологічних вимог. Це зумовлює те, що у різних професійних джерелах інформації оперують різними визначеннями і тлумаченнями поняття вимог.

Процес визначення та узгодження вимог у багатьох випадках представляється у вигляді деякої ділової гри. Суть гри полягає у визначенні зацікавленості сторін, які беруть у ній участь, встановленні норм і правил при імплементації інтересів кожної із сторін у готовому програмному продукті.

В процесі ділової гри можуть висловлюватися різні обмеження щодо того, які властивості є надзвичайно важливими, а які можна опустити, яким чином і у який спосіб буде забезпечено реалізацію потреб замовника, як досягти якості

кінцевого продукту. Отже з аналізу наукових публікацій стає зрозумілим, що формалізовані методи збору, накопичення та опису вимог є недосконалими, або й взагалі відсутніми у деяких випадках. Ось декілька прикладів інтерпретації вимог до програмного забезпечення:

– вимоги – це сукупність думок щодо функції та обмежень, які повинні бути реалізованими у системі;

– вимоги – це сукупність властивостей, якими повинна володіти система для того, щоб представляти собою цінність для кінцевого користувача;

– вимоги – це структурне представлення потреб у системі, а також того що і яким чином повинно бути імплементованим.

У відповідності до трактування міжнародним термінологічним глосарієм з комп'ютерних наук, вимога має включати у себе опис:

– обмежень та/або можливостей, що потрібні користувачу для виконання поставлених задач і досягнення мети;

– функціональності та умов використання системи на системному рівні для забезпечення виконання контракту між замовником та розробниками;

– правил і регламентованих норм, які відображають основні положення стандартів та інших формальних документів та використовуються при реалізації системи;

– функціональної здатності та обмеження середовища проектування та подальшої експлуатації системи.

Однією із загальноприйнятих класифікацій вимог до програмного забезпечення є така, яка визначає наступні види вимог:

– функціональні;

– системні;

– нефункціональні;

Потреби замовників і користувачів ПЗ, зазвичай, представляють у вигляді заданих умов, реалізація яких приводить до досягнення мети і розв'язання задач, що відображають бачення функціонального спектру та вигляду системи зацікавленою стороною. Потреби, або по-іншому вимоги користувачів, подаються

у форматі текстового опису сценаріїв використання системи, із застосуванням і представленням основних прецедентів, табличному вигляді, що відображає відгук системи на дію користувача.

Сукупність системних вимог відображають фактори зовнішнього впливу щодо функціонування програмного забезпечення у визначеному середовищі, а також накладають обмеження на використання апаратних і сторонніх програмних підсистем. Такі вимоги повинні бути врахованими при проектуванні архітектури ПЗ з врахуванням обмежень засобів візуального представлення інформації та функціональності системи в цілому. Фіксація і представлення системних вимог відбувається шляхом застосування спеціальних форм і шаблонів, які дозволяють спрогнозувати тип і вигляд вхідної та вихідної інформації у проектованій системі, а також забезпечити автоматизацію вимог.

Під вимогами до атрибутів якості ПЗ варто розуміти обмеження, які накладаються на мета рівні до властивостей і функцій, які є важливими для однієї або обох зацікавлених сторін Прикаладами атрибутів якості ПЗ можуть виступати вимоги щодо її міграції, цілісності, надійності, продуктивності, стійкості до відмов і збоїв. Функціональні вимоги представляють у вигляді переліку функцій і/або сервісів, якими повинна володіти і які повинна надавати програмна система у результаті її імплементації. До складу таких вимог можуть входити обмеження щодо структури і формату даних, а також опис умов поведінки системи при накладанні обмежень. Функціональні вимоги доцільно формувати у вигляді специфікацій – структурованого опису функцій і властивостей, що трактуються однозначно і не містить конфліктності, протиріч і виключень.

Основне призначення нефункціональних вимог полягає у визначенні умов виконання функціональності (наприклад, безпекові заходи при використанні БД, або авторизація прав доступу до ПЗ) в контексті, що безпосередньо не має зв’язку з імплементацією конкретної функції. Ці вимоги відображають потреби користувачів щодо обмежень на виконання функцій і характеризують особливості комунікації з іншими системами. Нефункціональні вимоги можуть накладати обмеження щодо продуктивності системи або її компонентів, захищеності

інформації та компонентів, а також ступінь забезпечення якості ПЗ з урахуванням нормативних документів та рекомендацій стандартів. Обмеження на функціональні та нефункціональні вимоги можуть задаватися і включати в себе різні кількісні показники з відповідними одиницями вимірювання: час очікування відповіді системи, обмеження щодо кількості користувачів, які працюють паралельно, імовірність безперебійної роботи та ін. До кінцевого програмного продукту висувають наступні нефункціональні вимоги:

– якість користувацького інтерфейсу та зручність використання;

– продуктивність, зокрема щодо пропускної здатності, час реакцій та відгуку системи, об’єму використовуваного дискового простору та оперативної пам’яті та ін.;

– надійність функціонування (відсутність помилок, дефектів та безперебійна робота);

– наявність зовнішніх програмних інтерфейсів для забезпечення взаємодії з іншими компонентами або сторонніми системами.

Фіксація та опис різних типів вимог повинен відбуватись з урахуванням рекомендацій стандартів, зокрема тих, що входять у серію ISO 250xxx та із застосуванням уніфікованого довідника понятійного і термінологічного апарату. Це дозволить однозначно трактувати сутності і властивості предметної області, а також чітко характеризувати призначення і функціональність системи. За допомогою специфікації вимог до ПЗ можна відобразити принципи та особливості взаємодії системи, що створюється, із зовнішніми платформами, системним програмним забезпеченням та інформаційними середовищами. Створення документованої специфікації вимог завершується на початку процесу проектування архітектури, однак може залежати від конкретної моделі життєвого циклу. Після цього відбувається процедура узгодження цього документу зі стороною замовника. Специфікація вимог використовується як керівництво з інструкцією дій для виконання задач з імплементації ПЗ на стадіях та у відповідних процесах ЖЦ. У випадку виявлення неузгодженості або

конфліктності вимог, виконується їхнє коригування та уточнення, а після цього вносять корективи у задачі і процес розробки ПЗ.

1.2. Аналіз методів та інструментів інженерії вимог

Як було зазначено раніше, процес визначення та аналізу вимог до ПЗ є фундаментальним процесом, від якості виконання якого залежить ефективність реалізації інших процесів на подальших етапах проектування та міра відповідності імплементованих властивостей очікуванням і сподіванням замовників. Для забезпечення якості процесів інженерії вимог необхідним є впровадження відповідних методів та інструментальних засобів, що становлять цілі технології розробки та аналізу вимог.

Етап визначення вимог умовно можна поділити на кілька підетапів, які включають в себе процедури:

– формулювання, визначення та аналізу потреб зі сторони замовника;
– трансформації і структуризації потреб у вимоги до ПЗ;
– аналіз, деталізація та адекватне і об’єктивне відображення вимог.

Процес безпосереднього виконання робіт з імплементації проекту програмного забезпечення починається з фази виявлення і фіксації вимог, що є найтрудомісткішим з поміж інших етапів ЖЦ. А це в свою чергу вимагає застосування системного підходу, який би забезпечив максимальну інформативність щодо функціональності майбутнього ПЗ та особливостей предметної області.

Імплементувати системний підхід на етапі розробки вимог потрібно спираючись на технології, які б у повній мірі забезпечили підтримку процесу формулювання і адекватного відображення потреб у ПС і дали б змогу проводити їх моніторинг протягом періоду виконання проекту.

Узагальнена та спрощена схема формулювання вимог до ПЗ представлена на рис. 1.2. Аналізуючи представлення (рис. 1.2) потрібно відмітити, що для формулювання вимог необхідно виконання трьох послідовних стадій, що

починаються з фази визначення користувацьких потреб і завершується специфікацією вимог до ПЗ.

Слід зауважити, що при проектуванні вимог незалежно від типу програмного продукту, який розробляється, початкова стадія завжди є сталою, оскільки перед тим, як щось реалізовувати потрібно володіти вхідними даними від замовника. На інших фазах проекту відбувається трансформація потреб у вимоги.

Технологія відображення потреб замовника у вимоги до ПЗ залежить від екосистеми CASE-засобів, якими володіє фірма-розробник. Це призводить до того, що на наступних етапах проекту одні і ті самі вимоги можуть інтерпретуватися по-різному.

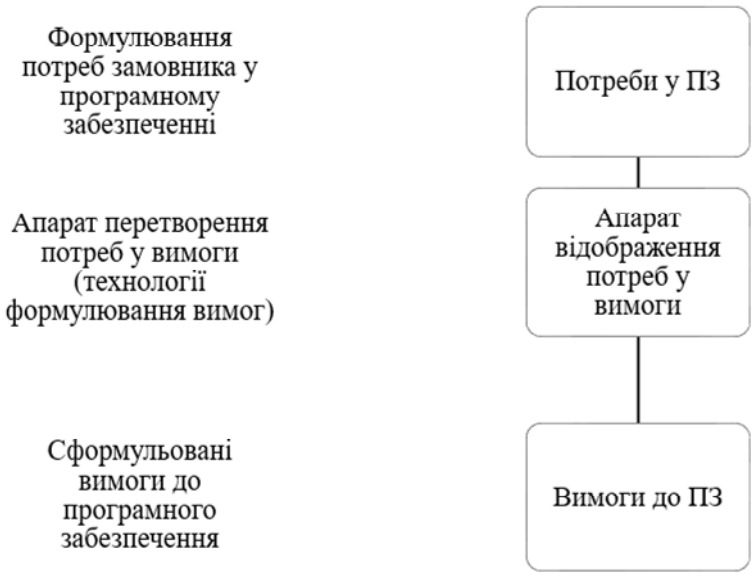


Рисунок 1.2 – Стадії розробки вимог до ПЗ

Сучасний етап розвитку технологій управління вимогами в індустрії програмного забезпечення передбачає використання трьох основних інструментів:

– технологічний стек, орієнтований на застосування специфікації вимог у відповідності до рекомендацій стандарту IEEE 830;

– технологія на основі методів і засобів структурного підходу;

– методи і засоби підтримки принципів об’єктно-орієнтованого програмування.

Однак, наведені вище технології містять ряд недоліків, зокрема це стосується недосконалості математичного апарату відображення вимог, які б задовольняли критеріям адекватності та однозначності трансформації потреб у вимоги до ПЗ. Як наслідок, це призводить до підвищення суб’єктивного впливу окремих учасників проекту на реалізацію властивостей продукту, вводить неоднозначність та не уніфікованість термінологічних особливостей.

При визначенні вимог доцільно використовувати і задавати критерії оцінювання їх якості та інтерпретації результатів. Це стосується як функціональних, так і нефункціональних вимог.

В загальному випадку вимоги до ПЗ повинні задовольняти наступним критеріям якості:

– зрозумілість та формалізованість;

– адекватність і вимірюваність;

– наявність кількісних метрик оцінювання.

Стандартом IEE 830 визначено структуру і зміст специфікації вимог, класифікація яких при цьому проводиться у відповідності до приналежності: або замовнику, або розробнику.

Критеріями якості специфікації вимог, згідно цього стандарту є:

– «коректність» – відображення адекватності потреб, висловлених замовником;

– «однозначність» – однакове трактування вимог усіма сторонами, які беруть участь в проекті;

– «повнота» – максимальне відображення усіх потреб замовника і властивостей ПЗ;

– «узгодженість» – відсутність конфліктних та суперечливих вимог;

– «впорядкованість» – наявність структури за критеріями важливості;

– «тестованість» – здатність вимог бути протестованими;

– «модифікованість» – можливість внесення та врахування змін у вимогах.

– «простежуваність» – здатність забезпечити трасування і комунікацію вимог в процесі виконання проекту.

Рекомендації і специфікацію вимог, визначену стандартом IEEE 830 можна представити у вигляді ієрархічної слабоформалізованої структури. Основні розділи специфікації вимог цього стандарту представлено у табл. 1.1.

Таблиця 1.1 –Характеристики розділів стандарту IEEE/ANSI 830

Розділ стандарту IEEE/ANSI 830	Короткий опис
Вступ	Містить мету створення програмної системи, визначає область застосування, перелік термінів, означень і скорочень, що використовуються при проектуванні системи
Загальний опис	Описують перспективи програмного продукту, тобто попередньо визначають системні, програмні, апаратні та комунікаційні інтерфейси, висувають вимоги до адаптації, операцій та обмежень ПС, визначають функції ПС, користувацькі характеристики і т.п..
Детальні вимоги	Визначають вимоги до інтерфейсів, функції ПС, вимоги до продуктивності, атрибути, якими

	характеризують систему.
Супроводжувальна документація	Містить специфікацію вимог та перелік документації на основі якої здійснювали проектування вимог до ПС.

Основним недоліком стандарту IEEE 830 є те, що у ньому не запропоновано метрик для оцінювання якості вимог, тому його застосування можливе лише при реалізації невеликих програмних проектів.

Ще одним підходом до формулювання та представлення вимог є підхід, що базується на використанні спец шаблонів і мов, які дають можливість описати вимоги у формальному вигляді і забезпечити поділ на структуру опису вимог і даних [6, 7].

Особливістю використання підходу шаблонів є те, що при представленні вимог, необхідно постійно формувати та резервувати набір окремих слів, або фраз, у вигляді узгодженого термінологічного довідника. При виконанні цих умов, такий спосіб визначення вимог стає доволі гнучким та ефективним засобом. При визначенні функціональних вимог доцільно використовувати резервацію слова «повинен», тоді прикладом шаблону може виступати наступна фраза: «Система <повинна> опис можливостей системи».

Як видно з прикладу, за допомогою шаблонів можна визначити деяку структуру та встановити послідовність слів при формулюванні вимог у текстовому вигляді.

Обмеження на вимоги також можна формувати за допомогою шаблонів. В основному вони задають умови на функціональних вимогах, або вимогах щодо продуктивності майбутньої системи.

Для прикладу, якщо маємо потребу, висловлену замовником у вигляді наступного твердження: «web-сайт повинен максимально швидко реагувати на дії користувача». У даному випадку не задано у явній формі обмеження щодо швидкості реакції, однак при застосуванні шаблонів дану вимогу можна

представити у наступному вигляді: «Система <повинна> <виконувана функція> <об’єкт> <не більше> значення <одиниці вимірювання>». Якщо говорити про конкретний екземпляр шаблону відносно такої вимоги до ПЗ, то він матиме вигляд: «Web-сайт повинен відреагувати на дію користувача не більше ніж через 2 с».

В загальному випадку структуру шаблону і його зв’язок з даними можна візуалізувати, як показано на рис. 1.3.

Шаблон
<Система> повинна
<виконувана функція>
на <об’єкт>
не більше ніж через
<значення продуктивності>
<одиниці вимірювання>
Вимога з використанням шаблону
<система>= web-сайт
<виконувана функція>= відреагувати
<об’єкт>= дія користувача
<значення продуктивності>=2
<одиниці вимірювання>=секунди

Рисунок 1.3 – Приклад вимоги у вигляді шаблону

Підхід з використанням шаблонів, представлений на рис. 1.4 забезпечує можливість генерації формального відображення вимоги у довільний момент часу. За необхідності існує також можливість внесення змін у вимоги, шляхом модифікації структури шаблона.

Недоліками підходу шаблонів є:

– постійна нестандартизованість шаблонів – характеризується відсутністю уніфікованої форми представлення вимог до ПЗ і суб’єктивною класифікацією даних;

– складність представлення нефункціональних вимог;

– неоднозначність сприйняття вимог різними учасниками;

– складність при оцінюванні рівня задоволення вимог в процесі їх атестації;

– відсутність загальних пропозицій при виборі метрик.

Складність впровадження технології шаблонів зумовлена також не універсальністю шаблонів, що передбачає необхідність їхнього створення при кожному новому проекті. Це негативно відбивається на тривалості виконання проектів, а також підвищує його вартість.

Окрім технологій формулювання вимог на основі рекомендацій стандарту і підходу шаблонів існує ще один спосіб виявлення вимог, який базується на системному моделюванні та формальних методах.

Підходи, які реалізує системне моделювання, засноване на використанні структурних та об’єктно-орієнтованих методів. При цьому можуть застосовуватися інструменти у вигляді діаграм потоків даних, ER-діаграм і діаграми станів, які формують основу структурного підходу до визначення вимог.

Приклад діаграми потоків даних представлена на рис. 1.4 і демонструє вимоги щодо реалізації електронного магазину.

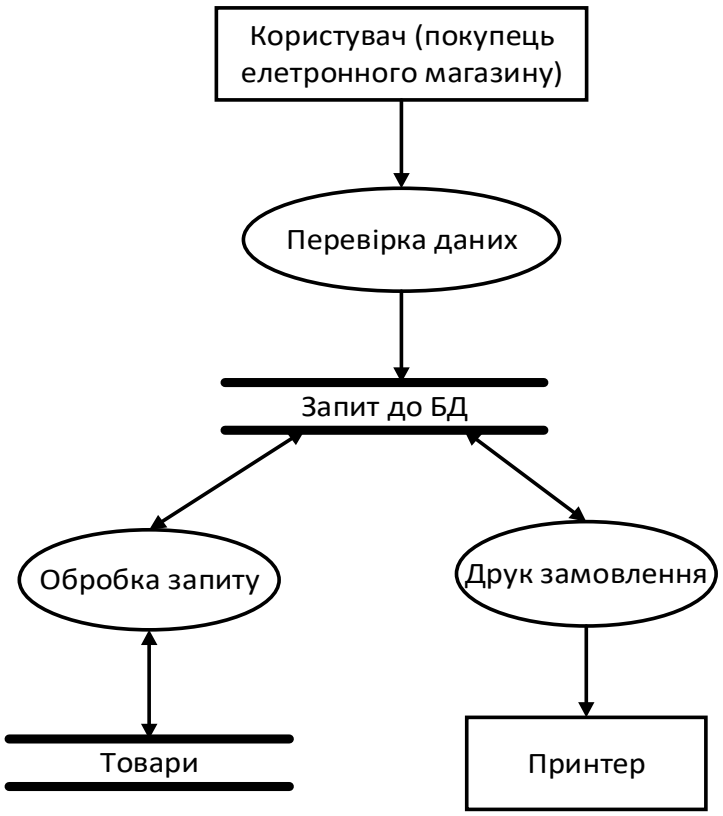


Рисунок 1.4 – Приклад діаграми потоків даних

Аналізуючи рис. 1.4, доволі складно виявити вимоги до програмного забезпечення, однак можна встановити сутності предметної області та потенційні зв’язки між ними.

До недоліків застосування діаграм потоків даних в контексті аналізу вимог до ПЗ належать недосконаліть методів забезпечення їх повноти, структурного представлення вимог та складність управління інформацією у відповідних потоках.

ER-діаграми дозволяють виявити сутності предметної області та встановити взаємозв’язки між ними, однак неможливо за їх допомогою представити комплексну картину і процеси, автоматизацію яких повинно забезпечувати програмне забезпечення, яке розробляється. Основне призначення діаграм станів полягає у фіксації процесу або об’єкту наявного в системі у визначений конкретний момент часу. Проте вони не забезпечують можливості безперервного моніторингу за станом процесів при функціонуванні системи. Навіть за умови одночасного застосування діаграм структурного підходу для формування вимог

до ПЗ не вдається забезпечити виконання критеріїв повноти, прозорості і структуризації вимог. Як наслідок, це не завжди коректно і правильно відображається на сприйнятті вимог замовником.

Об’єктно-орієнтований підхід при визначенні вимог до ПЗ реалізується за допомогою мови візуального моделювання UML, зокрема, use case діаграм. В продовження прикладу використання діаграми потоків даних структурного підходу, процес визначення товарів покупцем в електронному магазині за допомогою use case діаграми показано на рис. 1.5.

Рисунок 1.5 – Приклад використання use case діаграми

На основі побудованого use case сценарію (рис. 1.5) можна визначити функціональні вимоги до електронного магазину та врахувати ролі користувачів і їх функціональних можливостей.

У результаті аналізу як структурного, так і об’єктно-орієнтованого підходу встановлено, що їх доцільно застосовувати при виявленні лише функціональних вимог до ПЗ, однак вони не дають можливості відобразити нефункціональні вимоги, що значно обмежує сферу їх застосування.

1.3. Оцінювання підходів до представлення вимог до ПЗ

З метою кількісного вираження оцінок ефективності розглянутих підходів визначення вимог пропонується виконати експертне їх оцінювання за допомогою метрики наведених у стандарті IEEE 982.2. Ці метрики формують чотири групи критеріїв:

- показники якості;
- показники ефективності процесу верифікації вимог;
- кількісні міри ефективності аналізу вимог;
- критерії повноти вимог.

Показники якості вимог представляються критеріями, які можна виміряти в за допомогою яких встановлюється однозначність їх трактування, повноти, здатності до тестування, відслідковуваності, пріоритетності, атомарності та узгодженості.

В якості критерію ефективності перевірки вимог можна визначити метрику, яка кількісно виражає наприклад кількість опущених, або помилкових вимог, які відшукують протягом кожної години перевірки специфікації.

Прикладом метрики ефективності аналізу вимог є вартість щодо зміни та маніпулювання вимогами.

Оцінка метрики повноти дозволяє визначити рівень і здатність оперування вимогами у випадку, коли завершено збір та виявлення вимог. Результати застосування метрик оцінювання якості вимог наведені у табл. 1.2.

Таблиця 1.2 – Кількісне порівняння технологій формулювання вимог

№ п/п	Метрики	Технології проектування вимог		
		Технологія базована на стандарті IEEE 830	Технологія з використанням шаблонів	Технологія базована на використанні моделі якості
1	Однозначність	3	6	9
2	Завершеність	5	5	9
3	Тестованість	7	7	8
4	Елементарність	3	6	9
5	Швидкість зміни	8	4	6
6	Швидкість видалення	8	4	8
7	Швидкість додавання	7	4	8
Інтегральний показник		0,65	0,57	0,90

Інтегральний показник обчислювався як зважена сума:

$$I = \frac{1}{n \cdot b} \sum_{i=1}^{n} (W_i \cdot M_i), \tag{1.1}$$

де I - інтегральний показник оцінювання вимог;

W_i - вагові множники, які визначаються експертним шляхом для кожної з метрик;

M_i - величина метрики, яка оцінена експертами;

n $(n = 7)$ - загальна кількість оцінюваних метрик;

b $(b = 9)$ - максимальний бал, що визначається шкалою, обраною для оцінювання.

Візуалізація результатів оцінювання ефективності застосування технологій виявлення та представлення вимог показано на рис. 1.6.

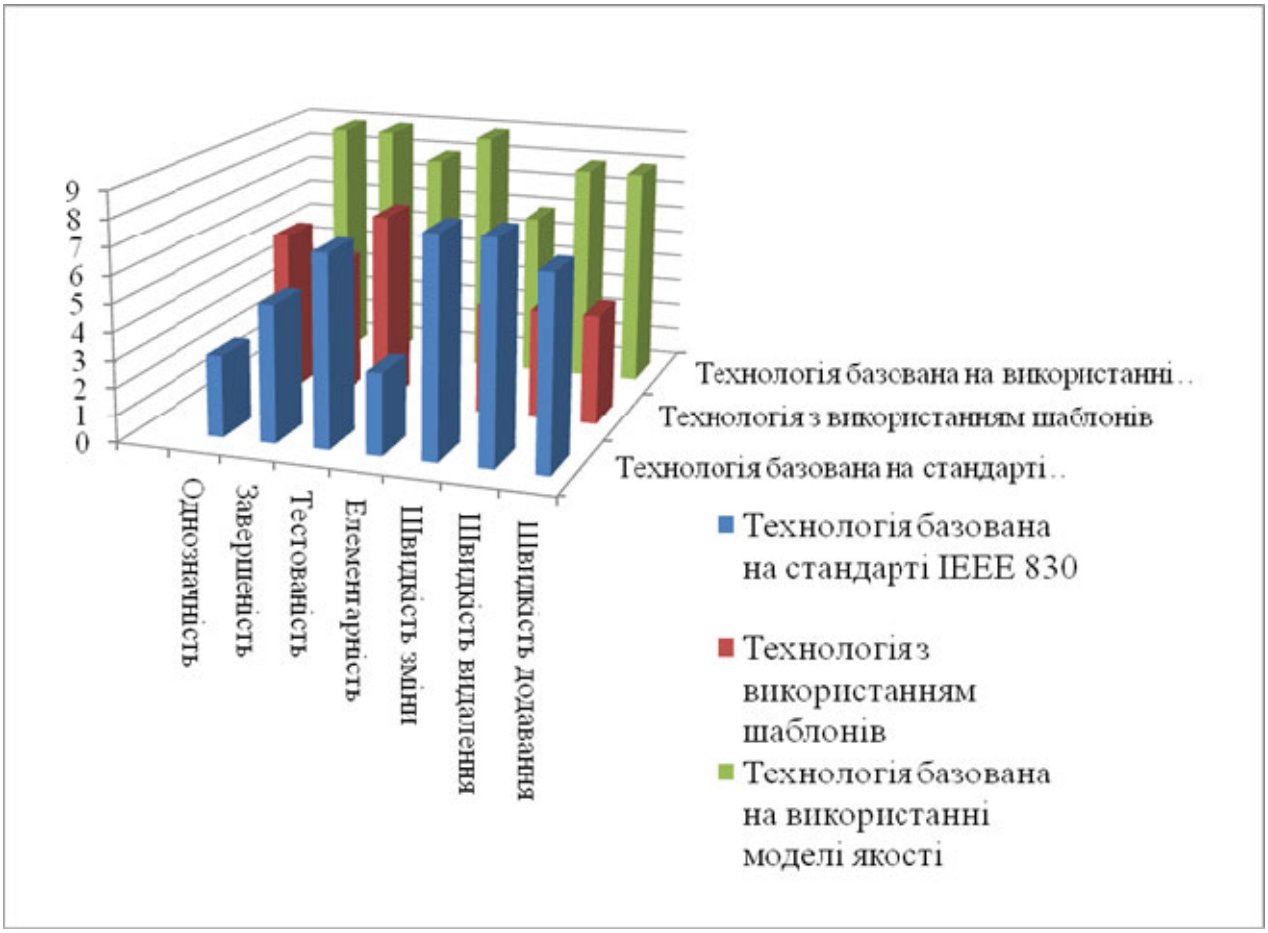


Рисунок 1.6 – Зведена діаграма оцінювання технологій визначення вимог до ПЗ

Виходячи з результатів оцінювання технологій визначення та аналізу вимог, представлених у табл. 1.2 та на рис. 1.6, а також рівня інтегрального показника, кращим варіантом представлення та управління вимогами є технологія, що передбачає використання моделі якості вимог. Перевагою такої технології є її ефективність, зумовлена застосуванням стандартизованих характеристик, які забезпечують однозначність та повноту опису вимог до ПЗ, а також здатність кількісно інтерпретувати оцінки якості виявлених вимог.

1.4. Висновки до розділу

1. На основі аналізу основних понять та класифікації вимог до програмного забезпечення визначено особливості процесів інженерії вимог та обгрунтовано актуальність дослідження моделей, методів і засобів управління зрілістю на стадіях життєвого циклу, що в перспективі забезпечить можливість їх оптимізації шляхом підвищення ефективності як самих процесів, так і якості готового програмного продукту.

2. Проведено аналітичний огляд методів та інструментів інженерії вимог у результаті якого визначено критерії якості при формулюванні вимог та підходи до реалізація цього процесу і встановлено, що на сьогодні найбільш ефективними є застосування підходів, які використовуються структуру специфікації рекомендовану стандартом IEEE 830, технологію використання шаблонів на основі структурного та об’єктно-орієнтованого підходів, а також технологію моделей якості.

3. Створено формалізовану схему щодо оцінювання підходів до формулювання вимог ПЗ, яка використовує експертні оцінки і дає змогу кількісно представити ефективність їхнього застосування в задачах забезпечення та управління зрілістю вимог на стадіях життєвого циклу. Встановлено, що кращим варіантом є використання моделей якості для представлення вимог до ПЗ.

4.

2 ОБГРУНТУВАННЯ ТА ФОРМАЛІЗАЦІЯ МОДЕЛІ УПРАВЛІННЯ ЗРІЛІСТЮ ВИМОГ

2.1. Аналіз та обґрунтування моделі зрілості управління вимогами до ПЗ

В якості базової моделі управління зрілістю вимог до програмного забезпечення запропоновано використати модель RMM («Requirements management maturity») Дж. Хеймана. Дана модель є розвитком більш загальної моделі, що використовується в індустрії програмного забезпечення – CMM («Software Capability Maturity Model»).

Особливістю цієї моделі є те, що вона може бути представлена у вигляді ієрархічної структури, де кожен вищий рівень ієрархії включає попередній. Така організація моделі дозволяє забезпечити безперервне вдосконалення процесу управління зрілістю вимог. На рис. 2.1. схематично показано структуру процесу управління зрілістю вимог.

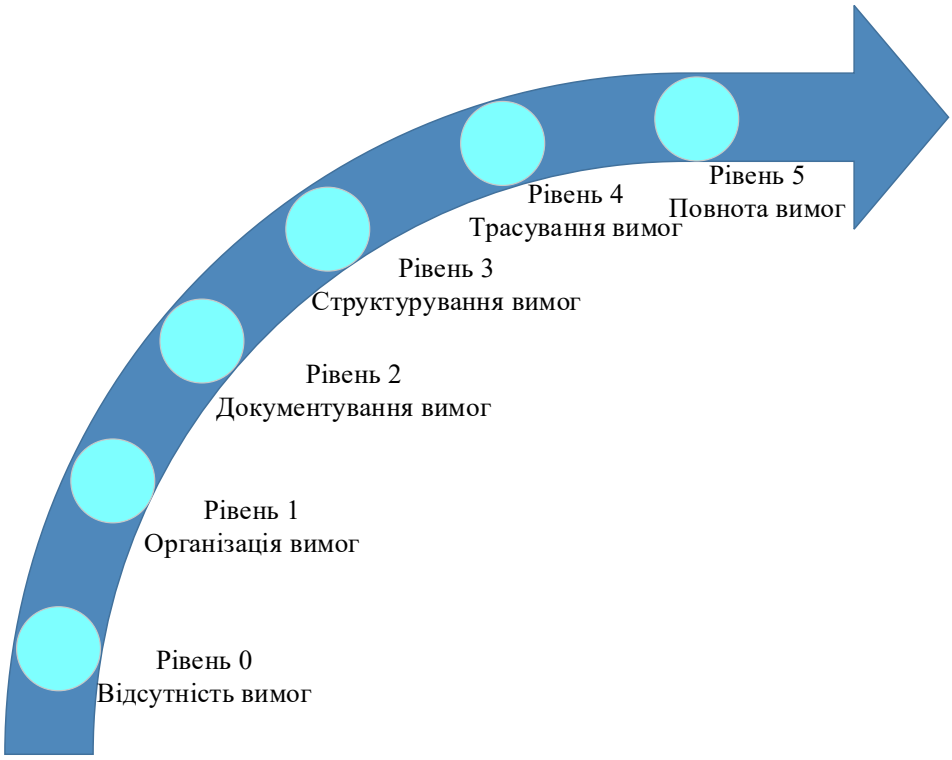


Рисунок 2.1 – Схематичне представлення рівнів моделі RMM

Структурно модель RMM складається з 6-ти рівнів зрілості, які дають можливість забезпечити ефективність впровадження процесу управління зрілістю вимог на різних етапах життєвого циклу.

Модель зрілості дає змогу розробниками забезпечити якість управління вимогами та удосконалити методи їх відстеження на етапах життєвого циклу.

Експериментально встановлено, що при впровадженні концепції постійного вдосконалення процесів забезпечення якості на основі моделі управління зрілістю процесів, перехід на кожен наступний рівень моделі RMM передбачає закріплення усіх методів та процедур попереднього та поточного рівнів. Залежність, що відображає відношення між якістю виконання проектів та часом при впровадженні моделі управління зрілістю вимог показано на рис. 2.2.

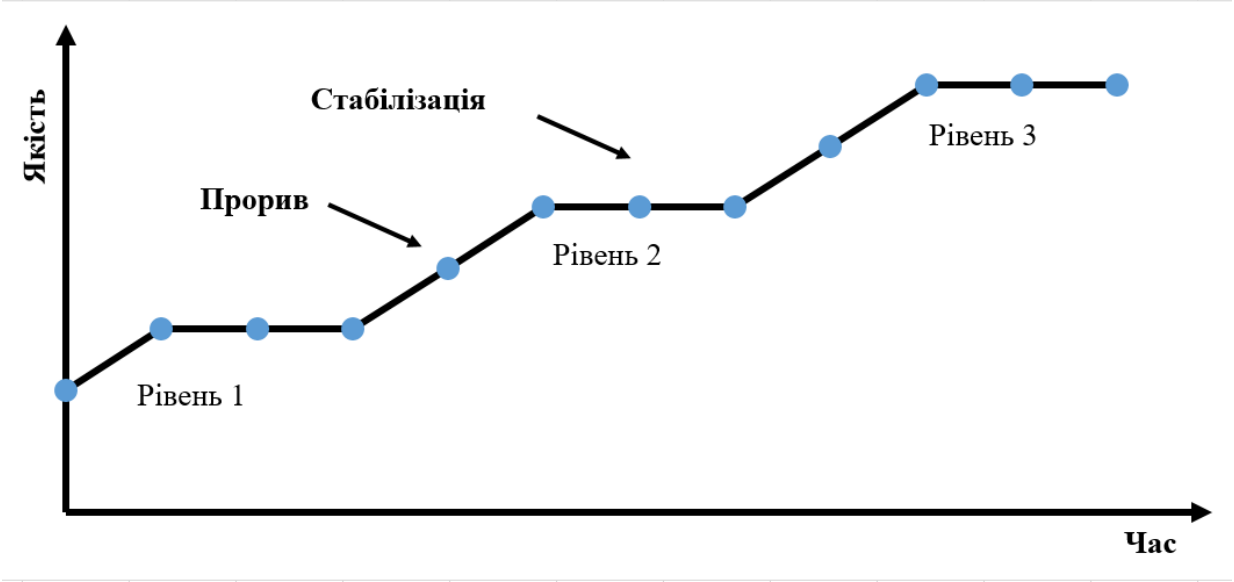


Рисунок 2.2 – Процедура управління процесом зрілості вимог та її вплив на якість виконання проекту

Проводячи аналіз графіка (рис. 2.2) того, як змінюється якість вимог у часі, можна висловити припущення про зростання рівня зрілості вимог та перехід команди розробників на більш якісний рівень. Це дозволить в перспективі розвинути еко систему CASE-засобів для управління вимогами та підвищити якість як вимог, так і продукту в цілому.

2.1.1. Характеристика нульового рівня зрілості вимог

Будь-яка команда розробників програмного забезпечення може відповідати нульовому рівню зрілості вимог, оскільки на цьому етапі спостерігається відсутність вимог як таких.

Причинами відсутності вимог щодо розробки ПЗ можуть бути наступні фактори:

– розробники не комунікують із стороною замовника для виявлення потреб у ПЗ, оскільки вони впевненні у правильності імплементації продукту;

– розробники економлять час на проектуванні системи, не звертаючи уваги на задачі виявлення, збору та документування вимог.

Виходячи з наведених вище причин, вимоги щодо реалізації властивостей ПЗ не документуються, з’являється різне бачення і трактування того, що повинно бути імплементовано командою розробників. При зміні учасників проекту реалізації програмного забезпечення, вимоги до нього можуть бути втраченими.

Ігнорування етапу збору та аналізу вимог призводить до того, що програмне забезпечення не задовольняє потребам замовника через те, що у ньому відсутній необхідний функціонал або ж навпаки – реалізовані функції, які не потрібні замовнику чи кінцевому користувачу.

Важливість подібних проблем залежить від замовника та ступеню критичності ПЗ. При розробці крупномасштабних чи корпоративних програмних продуктів такі проблеми можуть привести до провалу всього проекту.

Компанії і проектні команди для створення якісного ПЗ не можуть бути конкурентоспроможними на ринку, якщо у них не реалізовано процедури управління вимогами.

2.1.2. Аналіз особливостей першого рівня зрілості моделі RMM

Раніше було зазначено, що без реалізації процесів виявлення і фіксації відповідними засобами вимог до ПЗ неможливо реалізувати процес управління

ними на стадіях життєвого циклу. Перший рівень моделі RMM, на відміну від нульового, передбачає одержання потреб від замовника та формуванні документованої їхньої специфікації. Тобто, основна мета першого рівня при управлінні зрілістю вимог полягає у їхній документації.

Задокументовані вимоги є фундаментом при реалізації процесу керування зрілістю вимог. Окрім того, вони використовуються при проведенні та організації процесів тестування, впливають на вибір архітектурних рішень, оформлення документації щодо супроводу програмного забезпечення та інших організаційних процесів.

Наявність вимог до ПЗ також скорочує ризики, пов’язані із зміною персоналу і втратою вимог. При зміні учасників, які беруть участь у розробці проекту, наявна документація вимог до ПЗ дає змогу швидко зрозуміти, яку систему розробляє команда і якими властивостями володіє прототип системи. Можливість формування резервних копій задокументованих вимог знижує ризики їх втрати при виході з ладу апаратного забезпечення.

Однією з переваг документування вимог до ПЗ, орієнтованих на забезпечення якості кінцевого продукту, є узгодження документів із замовником. Проведемо більш детальний аналіз процесу «Документування» у моделі зрілості.

Процес добування і виявлення вимог до ПЗ може бути виконаний за допомогою різних методів. Серед ефективних підходів процесу визначення вимог найбільш ефективним з точки зору практики є:

– метод заснований на інтерв’ю;

– метод аналізу існуючої документації.

Процедура, що реалізує визначення вимог до майбутнього ПЗ може бути успішною лише тоді, коли відбувається комунікаційна взаємодія учасників проекту, зокрема, зі сторони замовника та представника команди розробників. Від команди розробників, зазвичай, виступає аналітик, основними функціями якого є формування сприятливого середовища з метою детального аналізу факторів, які впливають на розробку ПЗ. Перед тим як комунікувати зі стороною замовника

варто сформувати словник сутностей предметної області та зрозуміти лексичні значення термінологічних особливостей.

Одним з методів одержання вимог є проведення інтерв'ю. Цей метод орієнтований на одержання максимальної кількості інформації щодо бачення продукту замовником, яка функціональність є критичною для них, який зовнішній вигляд повинен мати інтерфейс користувача, чи передбачається взаємодія ПЗ з іншими системами і т.п.

Інтерв'ю проводиться у вигляді усного опитування за наперед визначеним планом. Відповіді замовника фіксуються засобами представників розробника.

Доволі часто буває ситуація, що майбутнє програмне забезпечення призначається для автоматизації бізнес діяльності замовника. При цьому, самі бізнес процеси вже є апробовані практикою, однак не є автоматизованими. У цьому випадку з'являється документація у вигляді регламентних документів, інструкцій та правил, якими керуються замовники і які є предметом аналізу для створення майбутнього програмного продукту. В такому випадку, аналітик отримує додаткову інформацію відносно того, які процеси необхідно автоматизувати і проведення інтерв'ю використовуються лише для узгодження вимог.

Окрім документації замовника, аналітики з команди розробників, можуть аналізувати подібні за функціональністю продукти, які вже існують на ринку та виявляти не очевидні вимоги до програмного забезпечення.

Після того, як вимоги до програмного забезпечення є задокументованими, необхідно провести їх перевірку на відповідність потребам замовника, тобто реалізувати процедуру валідації вимог.

Якість документу з вимогами (специфікації вимог) на першому рівні зрілості в більшості випадків не є високою, однак вони є відправною точкою для подальшого розвитку процесного підходу до управління вимогами.

Для перевірки специфікації вимог до програмного забезпечення, зазвичай, залучають експертів в даній області, або ж представників сторони замовника, які проводять рев'ю документації.

Експертне оцінювання дає змогу команді розробників одержати вимоги, які не суперечать процесам предметної області та які відповідають критерію якості вимог – правильність. Деякі метрики, для оцінювання результатів експертизи включають:

– об’єм тексту на кожній сторінці специфікації вимог;
– складність специфікації;
– ризики, пов’язані з помилками у специфікації;
– критичність матеріалів для успішного завершення проекту;
– кваліфікація аналітика, який написав специфікацію вимог.

Графік залежності кількості помилок на сторінку специфікації вимог від швидкості проведення експертизи наведено на рис. 2.3

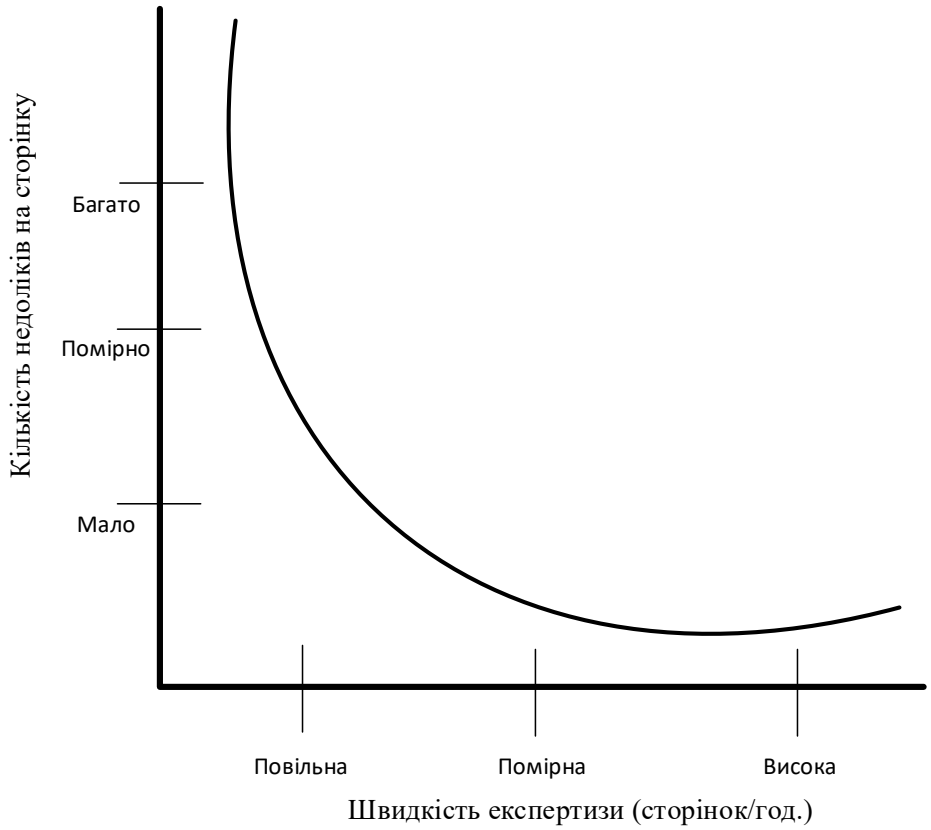


Рисунок 2.3 – Швидкість проведення експертизи

Виходячи, з аналізу графіку залежності кількості помилок від швидкості проведення експертизи, можна виділити основні правила проведення експертизи:

– 1 – 2 сторінки на годину вважається оптимальною швидкістю проведення експертизи для найбільш ефективного виявлення помилок у специфікації вимог;

– 2 – 4 сторінки на годину є рекомандваними з точки зору практики проведення експертиз.

У попередньому абзаці наведено середньо статистичні показники, щодо проведення експертизи специфікації вимог, однак при цьому необхідно враховувати і кваліфікацію експерта, складність предметної області та ін. Для забезпечення часових рамок виконання проекту, експертну оцінку варто починати проводити за наявності 10-20% специфікації вимог.

Узгодження специфікації вимог із замовником дає змогу забезпечити високий рівень відповідальності команди розробників, встановити межі еволюції програмного продукту, бюджету і часу імплементації проекту.

2.1.3. Характеристика другого рівня моделі зрілості RMM

Другий рівень моделі зрілості вимог передбачає виконання процедур і заходів для забезпечення зрозумілості та однозначного сприйняття сукупності вимог усіма учасниками проекту, а також визначення поведінки системи за різних умов використання.

Для одержання вимог, які відповідають таким критеріям якості, необхідно розробити уніфіковані шаблони документів, структурувати вимоги в єдину базу даних та забезпечувати процес фіксації змін у них. Оскільки, вимоги можуть використовуватись різними членами і групами команди, необхідно розмежувати права доступу до вимог. Затрати, які необхідні для переходу на другий рівень зрілості, пов’язані в основному з навчанням команди з новими методами роботи і додатковими перевірками вимог і специфікацій. Однак, ці затрати сприяють отриманню більш якісних вимог, що в перспективі знизить вартість і часові рамки виконання проекту. До затрат на другому рівні зрілості моделі RMM належать також і купівля інструментальних засобів управління вимогами до ПЗ.

До процесів, які притаманні другому рівню зрілості вимог належать:

– визначення вимог із застосуванням різних видів анкетування та «мозкового штурму» (brain storm);

– аналіз вимог;

– документування вимог;

– колективна валідація вимог з фіксацією виправлень;

– управління вимогами.

Процес визначення вимог на другому рівні моделі зрілості передбачає застосування методів анкетування та «мозкового штурму» разом із застосуванням методів виявлення вимог першого рівня зрілості.

Метод анкетування ефективно застосовувати у випадках:

– аналітику складно зустрітися із замовником через значну територіальну віддаленість;

– складно визначити кінцевих користувачів системи і виділити серед них таких, з якими можна провести інтерв’ю;

– маркетологами досліджується ринок, і система розробляється як комерційний продукт;

Анкетування рекомендовано проводити в комплексі з інтерв’ю, що дає змогу більш детально описати предметну область і потреби кінцевих користувачів. Основною вимогою до анкет є простота розуміння запитань, відповіді на які повинні бути відкритими. Оскільки, анкетування є процесом, що передбачає розповсюдження анкет, їхній збір та аналіз, то проводити його необхідно якомога раніше.

Застосування «мозкового штурму» на етапі виявлення, аналізу і документації вимог сприяє виявленню неочевидних вимог, встановленню зв’язків між вимогами та забезпечує несуперечність вимог до ПЗ.

Другий рівень моделі зрілості RMM передбачає виконання процесу аналізу вимог, що включає в себе уточнення вимог та їх документування. Перш за все, вимоги аналізуються на повноту та несуперечність. Наступний крок полягає у перевірці вимог та їх документуванні, що дає змогу виявити конфліктність вимог.

У випадку виявлення конфліктності вимог, аналітику потрібно повернутись до етапу виявлення вимог, довизначити вимоги та усунути їх конфліктність.

Документування вимог до ПЗ на другому рівні моделі зрілості базується на використанні методів спільного доступу членів команди до документації. Вимоги до документації висуваються таким чином, щоб специфікація вимог була зрозумілою, простою у використанні та уніфікованою. Відповідність специфікації вимог критеріями якості запропоновано реалізовувати шляхом застосування шаблонів. При цьому передбачається застосування шаблонів синтаксичних конструкцій і шаблонів форматування.

Процес перевірки вимог на другому рівні зрілості передбачає реалізацію двох підпроцесів: колективна перевірка та формування документу із зауваженнями.

Колективна перевірка вимог представляє собою різновид експертного оцінювання, у якому беруть участь не тільки аналітик і експерт у предметній області, а й інші члени команди. Як приклад, системний архітектор або «team lead» перевіряють вимоги на здатність до реалізації із застосуванням прийнятих у команді технологій, платформ, та інших обмежень. Тестувальники перевіряють вимоги на можливість їх подальшого тестування.

Даний процес забезпечує досягнення прийнятних рівнів якості вимог за критеріями повноти і здатності до перевірки.

Формування документу із примітками, при перевірці вимог, є важливим інструментом досягнення якості вимог, оскільки, при колективній перевірці специфікації вимог беруть участь багато учасників. Основна задача аналітика, який відповідає за якість специфікації вимог, полягає у формуванні зведеної таблиці. Структура зведеної таблиці повинна містити вимоги, примітки до вимог та інформацію про учасника, який сформував примітку.

Управління змінами на другому рівні моделі зрілості передбачає формування бази даних (БД) вимог та реалізацію процесу управління версіями вимог.

Вимоги можуть зберігатись не тільки у текстовому вигляді, але й у базі даних. Вагомою перевагою зберігання вимог у БД є можливість їх наступного використання на інших етапах розробки ПЗ. Сучасні інструментальні засоби для роботи з вимогами надають можливості їх зберігання у реляційних базах даних. Серед інструментів зберігання та управління вимогами до ПЗ можна виділити продукти фірми IBM – IBM DOORS, IBM Rational Request PRO.

При роботі з великою кількістю вимог до ПЗ, особливо на етапі їх уточнення і доопрацювання, необхідно контролювати історію версій вимог і специфікацій. Версійність дає змогу відслідковувати зміни у вимогах, причини внесення таких змін та можливість повернення до попередніх версій вимог. На ринку програмних продуктів існує велика кількість систем управління версіями, зокрема, Sybase PowerDesigner, Borland CaliberRM, IBM Rational RequisitePro та ін.

2.1.4. Третій рівень моделі зрілості вимог програмного забезпечення при проектуванні комп’ютерних систем

Третій рівень зрілості вимог до програмного забезпечення характеризується плануванням процесу керування якістю вимог, класифікації вимог за типами відносно подібності ознак. При записі вимог у БД ідентифікатор відіграє роль службової інформації та класифікатора вимог. При виділенні класів вимог проводиться їх структурування, за рахунок чого досягається певний рівень якості вимог.

Планування процесу керування вимог дає змогу формалізувати сам процес управління вимогами, оскільки, чітко визначає типи вимог, їх атрибути, набір необхідної документації і задачі, пов’язані з управління вимогами.

Типи вимог, в залежності від робочого процесу конкретної організації, класифікують як функціональні, нефункціональна та системні. Функціональні вимоги визначають функції, які повинні бути реалізовані у програмному продуктів. Нефункціональні вимоги – вимоги щодо якості функцій та обмежень

на них, зокрема вимоги до надійності, продуктивності, зручності у використанні та ін. До системних вимог належать вимоги до середовища, в якому буде використовуватись програмний продукт і відповідно враховує особливості операційних систем і додаткового програмного забезпечення.

Атрибути вимог є основною для створення якісних вимог. За допомогою атрибутів реалізується управління вимогами, відстеження стану вимог, одержується додаткова інформація та текстовий опис вимог. При використанні атрибутів спрощується опрацювання вимог, пошук, вибірка та сортування.

Для виявлення вимог, їх аналізу та документування можуть застосовуватись підхід на основі діаграм прецедентів (Use Case Driven Approach) і прототипування.

При розробці програмних продуктів на основі об’єктно-орієнтованого підходу, варіанти використання забезпечують комплексне уявлення про продукт з точки зору взаємодії різних користувачів з системою, тим самим сприяють виявленню функціональних вимог. Такий підхід передбачає раннє залучення користувачів і замовників програмної системи у процес її розробки, що сприяє більш точному розумінню функцій, які повинні бути реалізованими.

На основі варіантів використання в подальшому будуються моделі аналізу вимог та проектування архітектури. Побудова моделей і сценаріїв використання вимагає наявності значного практичного досвіду в аналітиків.

Принципи розробки варіантів використання передбачають створення системної діаграми на початковому етапі визначення вимог, а в подальшому проводиться її декомпозиція з детальним опис функцій, які повинні виконувати система.

Системна діаграма варіантів використання дає змогу визначити межі системи та основних користувачів, які в перспективі будуть взаємодіяти з нею. Далі моделюються варіанти використання, зазвичай, у вигляді «дієслово+іменник».

Наприклад, вимога користувача: «Користувач повинен за допомогою системи оплачувати покупки в інтернет магазині», можна записати наступним

чином: «Оплата покупки в інтернет магазині» або «Оплатити покупку в інтернет магазині».

Якщо система є крупномасштабною і охоплює багато функціональних вимог, то системну діаграму зображають у вигляді пакетів варіантів використання. Приклад діаграми пакетів використання наведено на рис. 2.4.

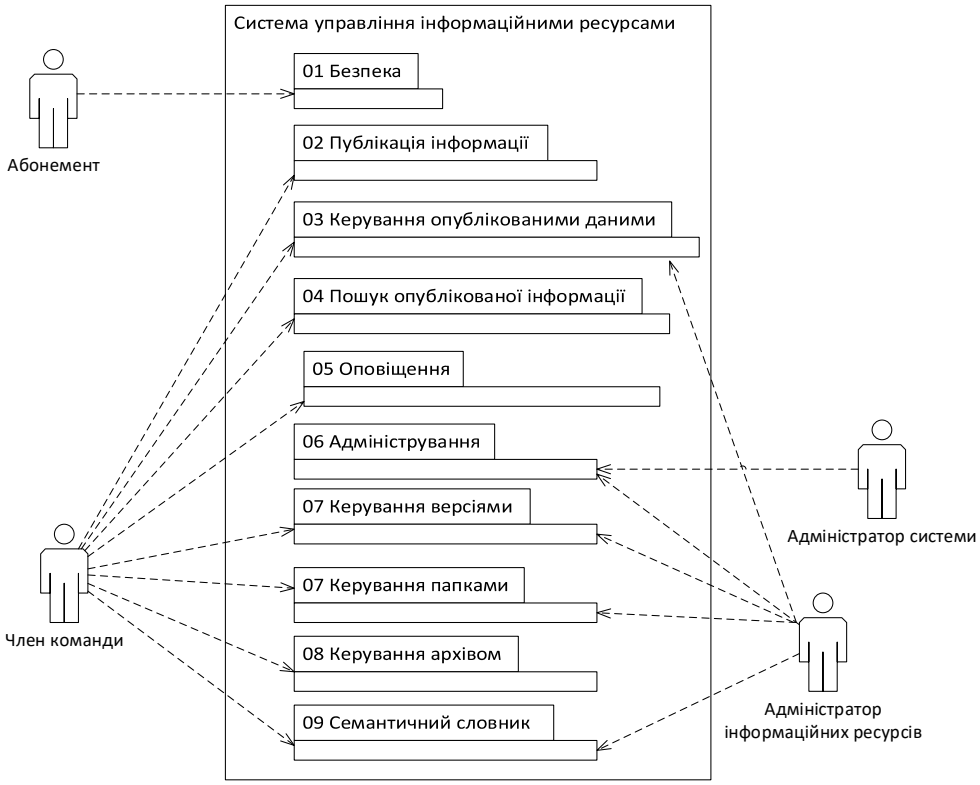


Рисунок 2.4 – Приклад системної діаграми у вигляді пакетів варіантів використання

У випадку реалізації крупномасштабного проекту, у команді необхідно виділити учасників, які б займались збором та аналізом вимог до кожної підсистеми. На рис. 2.4 визначено границі системи та функції, які необхідно реалізувати. Користувачі майбутньої системи знаходяться за її межами.

Кожна функціональна область повинна бути представлена у вигляді окремої діаграми прецедентів (рис. 2.5).

Кожен варіант використання повинен бути описаний у вигляді сценарію. Система на рівні абстракції варіантів використання представляє собою «чорну

скриньку», тобто при розробці сценарію описується, що повинна робити система, а не спосіб як досягнути функціональності.

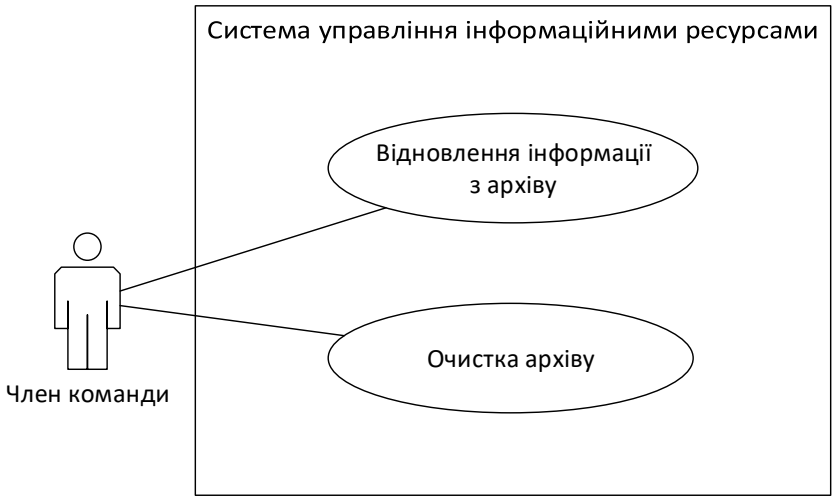


Рисунок 2.5 – Приклад діаграми прецедентів

В залежності від технічних знань замовника чи користувача системи сценарій варіантів використання можна описувати у текстовому вигляді, або ж у вигляді діаграм видів діяльності та послідовностей.

Текстовий варіант опису сценаріїв використання може бути представлений як простий опис, у табличному або структурованому вигляді. Найбільш прийнятним є представлення варіанту використання у структурованому шаблонному вигляді, що дає змогу зрозуміти, яку функцію або сукупність функцій необхідно реалізувати у системі.

Поряд з Use Case діаграмами варто використовувати прототипування системи, користувацьких інтерфейсів та функціональних можливостей подібних систем.

Для аналізу вимог на третьому рівні моделі зрілості вимог необхідно структурувати вимоги і для кожної з них задати значення атрибутів. Це дозволить одержати зрозумілий і керований список вимог.

Структурування вимог виконується на етапі аналізу паралельно з виявленням і документуванням вимог до ПЗ. Структуроване подання вимог дозволяє:

- мінімізувати загальну кількість вимог;
- краще зрозуміти велику кількість інформації;
- знайти набори вимог, які відносяться до визначеної тематики;
- виявити неточності і повтори у вимогах;
- забезпечити несуперечність вимог;
- керувати етапами реалізації;
- оцінити вимоги з точки зору часу і вартості реалізації;
- визначити малоінформативні вимоги;
- повторно використовувати вимоги у наступних проектах.

Структурувати вимоги можна шляхом розбиття документу на розділи, або на основі моделей аналізу вимог (пакетів вимог на системних діаграмах). Структуризація вимог забезпечує їх відповідність критеріям якості в контексті здатності до модифікації.

Атрибути вимог є джерелом для встановлення пріоритетів вимог і повинні бути задані на початку проектування програмного забезпечення. На основі атрибутів можна проводити трасування вимог на різних етапах проектування та оцінювати якість їх реалізації, визначати більш пріоритетні вимоги. Однак кількість атрибутів не повинна бути надмірною, а лише чітко і конкретно описувати ту чи іншу вимогу. При цьому атрибути формулюють таким чином, щоб можна було кількісно виміряти їх значення.

Для документування вимог можна використовувати підхід, який базується на шаблонах. Такий підхід має ряд переваг в порівнянні з простим текстовим описом, оскільки дає змогу певним чином стандартизувати представлення вимог і провести їх класифікацію. Використання шаблонів вимог дозволяє по різному описувати різні вимог.

Для кожного проекту аналітик може розробляти свої шаблони, які в подальшому можна буде використання в наступних проектах. Маючи базу шаблонів, вимоги швидко структуруються і забезпечується зрозумілість функцій, забезпечення виконання яких повинна система. Процес створення вимог на основі шаблонів умовно можна розділити на два етапи:

– вибір найбільш відповідного шаблону з набору існуючих;

– заповнення порожніх полів у шаблоні конкретними даними.

На третьому рівні зрілості процесу керування вимогами окрім експертних оцінок потрібно використовувати контрольні листи і рекомендації. Правильно складені питання в контрольних листах, дають змогу розробникам перевіряти вимоги на наявність неточностей, помилок і не врахованих вимог.

Рекомендації дозволяють зрозуміти суть процесу опрацювання вимог. На даному рівні зрілості рекомендується створювати контрольні листи і рекомендації для усіх етапів процесу і їхніх артефактів (документів та моделей).

Контрольні листи (checklists) використовуються для забезпечення якості вимог. Вони представляють собою список питань, які член команди розробників ставить перед собою при перевірці вимог. Запитання повинні бути складені таким чином, щоб вони могли повністю покрити задачу або артефакт, що перевіряється.

Рекомендація (guideline) представляє собою додаткову інформацію про те, як працювати з певним конкретним документом, моделлю і задачею. До рекомендацій часто відносять навчальні матеріали, пояснення, коментарі , які дають змогу аналітику більш детально зрозуміти яка вимога і яким чином буде реалізована. Наявність рекомендацій забезпечує скорочення часу на навчання персоналу, оскільки рекомендації є завжди доступними у базі знань відповідного проекту.

У випадку досягнення перших трьох рівні зрілості управління вимогами, команда розробників матиме змогу встановлювати і відстежувати відношення між вимогами до програмного забезпечення.

2.1.5. Четвертий рівень моделі зрілості вимог

Мета встановлення відношень між вимогами полягає у можливості в подальшому відстежувати зміни у вимогах, визначати вплив одних вимог на інші, забезпечувати ідентифікацію надлишкових вимог та неочевидних вимог. Встановлення таких зв’язків дозволить підвищити якість процесу управління

вимогами шляхом застосування строгих правил побудови залежностей між ними, аналізу повноти та впливу одних вимог на інші.

Основним завданням четвертого рівня моделі зрілості процесу управління вимогами до програмного забезпечення комп'ютерних систем є визначення і встановлення зв'язків між вимогами. При виявленні, зміні і перевірці вимог можуть виникати протиріччя і конфлікти із замовником або всередині команди розробки. Для розв'язання таких ситуацій пропонується організовувати семінари та обговорення з фокус-групами.

При розробці комп'ютерних систем і їх складових формується велика кількість вимог, які можуть утворювати ієрархію. Ієрархія може починатись на рівні бізнес-моделювання і постановки цілей, включати потреби користувачів і закінчуватись деталізацією вимог до комп'ютерної системи та її компонентів.

На попередньому рівні моделі зрілості проаналізовано типи вимог, які представляють собою різні рівні абстракції. Класифікація типів вимог в ієрархічну структуру і встановлення відношень між ними дає змогу спростити роботу з великою кількістю вимог і забезпечити їх трасування.

На практиці аналітики використовують кратність при визначенні відношень між вимогами. Кратність дозволяє задавати правилами, які визначають кількість залежних між собою вимог. Наприклад, правило може полягати в тому, що для кожної бізнес вимоги повинна бути визначена одна ключова можливість реалізації, і для кожної ключової реалізації повинні бути визначені декілька варіантів використання. Кратність відношень між вимогами повинна визначатись аналітиком і бути зафіксованою в план керування вимогами.

Здатність відслідковувати відношення між вимогами називається простежуваністю і визначається відношеннями трасування між вимогами. Простежуваність забезпечує здатність розуміння того, яким чином зміни однієї вимоги можуть впливати на інші вимоги, а також допомогти у визначенні відносної повноти вимог. Правила трасування вимог також повинні бути включеними у план управління вимогами.

Аналіз впливу однієї вимоги на іншу, перш за все, призначений для того, щоб керувати вимогами. За допомогою зв’язків між вимогами аналітик приймає рішення про зміни у вимогах. Наприклад, якщо замовник бажає змінити ключову вимогу і це спричинить зміни у декількох варіантах або сценаріях використання, то необхідно враховувати позитивні і негативні фактори такої зміни. Аналіз впливу щодо зміни вимог найпростіше виконувати за допомогою матриці і дерева трасування, яке показує зв’язки між вимогами.

Застосування типових рішень при аналізі і документуванні вимог дає змогу суттєво підвищити їх якість, проте потребує високого рівня кваліфікації спеціалістів (членів команди).

2.1.6. П’ятий рівень моделі зрілості вимог

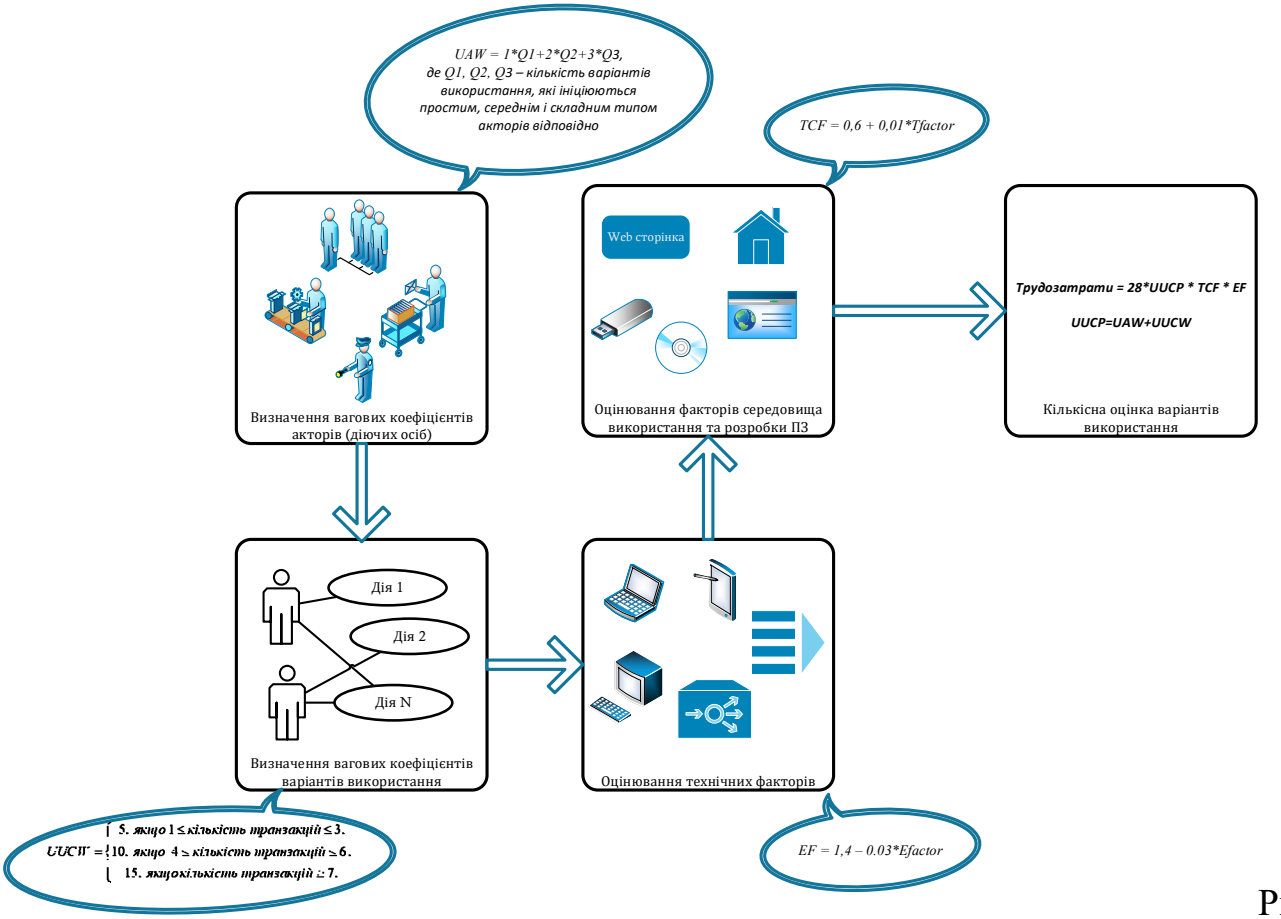
Основна мета п’ятого рівня моделі зрілості вимог полягає у тому, що вимоги використовуються не тільки для узгодження із замовником, а й використовуються на наступних ітераціях реалізації проекту.

Доволі часто має місце, коли вимоги призначені для того, щоб отримати згоду замовника про майбутні функції комп’ютерної чи програмної системи і в подальшому не пов’язані із процесом розробки. Це призводить до того, що вимоги в процесі створення системи «застарівають» і як результат – реалізований продукт не задовольняє потреби та очікування сторони замовника. Передбачається, що на п’ятому рівні виконується інтеграція автоматизованого засобів опрацювання вимог з CASE-засобами, які виконують функції підтримки процесів безпосереднього проектування та імплементації системи. Для досягнення подібних цілей необхідно розробити та впровадити в процес управління вимогами показники оцінювання вимог для управління проектом, забезпечити трасування вимог на етапи проектування і тестування, а також запровадити систему управління змінами.

На п’ятому рівні моделі зрілості важливим для створення ПЗ, що відповідає очікуванням замовника, необхідно процес розробки побудувати таким чином, щоб команда використовувала вимоги, як основні вхідні дані.

Основною задачею системних архітекторів в даному випадку є забезпечення повноти реалізації вимог в архітектурі програмного забезпечення комп’ютерної системи. Як приклад, при застосуванні технології RUP (Rational Unified Process) варіанти використання враховуються при аналізі вимог та при проектуванні архітектури і модулів системи. Встановлення зв’язків між вимогами і компонентами проектування архітектури, при цьому, відіграє важливу роль. Трасування або комунікація вимог такого роду дає змогу простежити реалізацію системи, починаючи від бізнес вимог і закінчуючи класами і програмним кодом.

Кількісна оцінка варіантів використання розроблена Г. Карнером як спосіб вимірювання розміру програмного забезпечення по кількості, розміру і складності варіантів використання програмного забезпечення. Процес обчислення оцінок прецедентів наведено на рис. 2.6.



Рис

унок 2.6 – Кількісна оцінок варіантів використання

Процес визначення трудозатрат починається з визначення вагових коефіцієнтів акторів, які ініціюють варіант використання. Ваговий коефіцієнт залежить від типу актора, характеристика і величина якого наведені у табл. 2.1.

Далі розраховується інтегральний ваговий коефіцієнт акторів за формулою:

$$UAW = 1*Q_1 + 2*Q_2 + 3*Q_3 \qquad (2.1)$$

де, Q_1, Q_2, Q_3 – кількість варіантів використання, які ініціюються простим, середнім і складним типом акторів відповідно.

Таблиця 2.1 – Вагові коефіцієнти акторів

Тип	Опис	Ваговий коефіцієнт

Простий	Зовнішня система, що взаємодіє через прикладний програмний інтерфейс	1
Середній	Зовнішня система, що взаємодіє через протоколи, такі як TCP/IP або користувач, який взаємодіє через інтерфейс на основі командної стрічки	2
Складний	Користувач, що взаємодіє з системою через графічний користувацький інтерфейс	3

На наступному кроці потрібно визначити вагові коефіцієнти для варіантів використання. Оцінювання даних коефіцієнтів наведена у табл. 2.2. Під терміном «транзакція» в даному випадку розуміють загальну кількість потоків подій (основних та альтернативних).

Таблиця 2.2 – Вагові коефіцієнти варіантів використання

Тип	Кількість транзакцій	Ваговий коефіцієнт
Простий	1-3	5
Середній	4-7	10
Складний	7 і більше	15

Формально ваговий коефіцієнт для одного варіанту використання ($UUCW_i$) можна записати наступним чином:

$$UUCW_i = \begin{cases} 5, \text{ якщо } 1 \le \text{кількість транзакцій} \le 3, \\ 10, \text{ якщо } 4 \le \text{кількість транзакцій} \le 6, \\ 15, \text{ якщо кількість транзакцій} \ge 7, \end{cases} \quad (2.2)$$

де i – порядковий номер варіанту використання.

Далі розраховується інтегральне значення вагового коефіцієнта для усіх варіантів використання. Для цього необхідно оцінити вагу кожного варіанту використання і знайти їх суму

$$UUCW = \sum_{i=1}^{N} UUCW_i \tag{2.3}$$

де N – кількість варіантів використання

Наступним кроком при визначенні трудозатрат є знаходження суми інтегрального показника ваги акторів та інтегрального показника варіантів використання. У результаті цього одержуємо не усереднену кількісну оцінку варіантів використання:

$$UUCP = UAW + UUCW \tag{2.4}$$

Розрахунок впливу зовнішнього середовища і технічного фактора в загальному розрахунку трудозатрат визначають за формулами 2.5 і 2.6 відповідно.

$$EF = 1{,}4 - 0{,}03 \cdot Efactor \tag{2.5}$$

$$TCF = 0{,}6 + 0{,}01 \cdot Tfactor \tag{2.6}$$

У формулах 2.5 і 2.6 *EFactor* і *TFactor* є коефіцієнтами, які характеризують вплив зовнішнього середовища і вимоги до характеристик якості. Значення цих коефіцієнтів в залежності від зовнішнього середовища та технічного забезпечення можна взяти із таблиць наведених в [11].

На основі одержаних значень можна розрахувати трудозатрати в людино-годинах, які необхідні на реалізацію варіанта використання у вигляді програмного коду. Формула для розрахунку трудозатрат матиме вигляд:

$$Трудозатрати = 28 \cdot UUCP \cdot TCF \cdot EF \quad (2.7)$$

Використовуючи такий підхід до розрахунку трудозатрат керівник проекту може розрахувати терміни виконання проекту та його бюджет, а також сформувати команду розробників.

При інтеграції з іншими інструментальними засобами, моделі вимог можуть розроблятися в їхньому середовищі. Відповідно це дозволить генерувати звіти на основі відповідних моделей і документацію з вимогами. Сучасні інструментальні засоби проектування і розробки програмного забезпечення дають можливість отримати майже всю необхідну інформацію, яка стосується моделей вимог, моделей аналізу та моделей проектування.

Для управління вимогами на п’ятому рівні моделі зрілості передбачається використання системи керування змінами, що орієнтована на те, що жодна зміна у вимогах не може відбутись без аналізу і затвердження.

Впровадження процесу управління вимогами до ПЗ є доволі складною і неоднозначною задачею. Для спрощення, зниження порогу входу та підвищення якості вимог запропонований підхід поетапного впровадження, який базується на моделі RMM. Кожен етап при цьому відображає відповідний рівень зрілості.

2.2. Формалізація моделі RMM

Провівши детальний опис кожного рівня моделі зрілості (Requirements Maturity Model) і відповідних процесів, формалізуємо модель управління зрілістю вимог у термінах теорії множин. В загальному випадку модель зрілості можна записати таким чином:

$$RMM = \{L_N \{L_{N-1} \{L_{N-2} \{ \ldots \{L_0\} \ldots \}\}\}\} \quad (2.8)$$

де L – рівень зрілості моделі вимог, N – кількість рівнів моделі зрілості.

Згідно обґрунтованої моделі, кількість рівнів зрілості дорівнює 6. L_0 – нульовий рівень зрілості вимог, що передбачає їх відсутність в плані хаосу та наявності лише частини потреб замовника. Вимоги до програмного забезпечення на цьому рівні відсутні.

Для забезпечення повноти потреб замовника або користувачів комп'ютерної системи на даному рівні пропонується автоматизувати процес збору шляхом підключення таких джерел даних як Skype, E-mail, та інші документи MS Office. Це дозволить зберігати в єдину базу даних інформацію про потреби користувачів у програмному забезпеченні і тим самим швидше перейти на другий рівень моделі зрілості вимог.

Формально перший рівень моделі зрілості вимог можна записати у вигляді:

$$L_1 = \{Doc_i, ExpMark_i, \{L_0\}\}, i = 1..T \tag{2.9}$$

де Doc – множина документів, які описують вимоги до ПЗ, $ExpMark$ – експертні оцінки документів,

T – кількість документів, що описують вимоги на першому рівні моделі зрілості.

Потреби у програмному забезпеченні представляють у вигляді:

$$R_c = \{P_i, C_{ik}\},\ i = \overline{1..I},\quad K = \overline{1..M_i}, \tag{2.10}$$

де P_i – потреби користувача;

C_{ik} - обмеження на потреби;

I– кількість потреб замовника;

K – кількість обмежень на потреби.

Другий рівень моделі зрілості вимог до програмного забезпечення орієнтований на уточнення вимог до програмного забезпечення шляхом

застосування різних методів та формування відповідної бази даних вимог. Його формально можна представити наступним чином:

$$L_2 = \{R_i^c, MClar_{ij}, RClas_{ik}, RVer_{im}, \{L_1\{L_0\}\}\} \quad (2.11)$$

де R_i^c – вимоги замовника до ПЗ, $i=1..B$, B – кількість вимог на другому рівні моделі зрілості;

$MClar_{ij}$ – методи уточнення вимог (мозковий штурм, анкетування, уточнення, колективна перевірка і т.п.), $j=1..S$, S – кількість методів для уточнення вимог;

$RClas_{ik}$ – класи до яких належать вимоги R_i, $k=1..K$, K – кількість класів вимог до ПЗ;

$RVer_{im}$ – версії вимоги до програмного забезпечення для контролю внесення у неї змін, $m=1..M$, M – кількість версій вимоги.

Для класифікації вимог пропонується використати два класи, як показано нижче:

$$RClas = \{Func, NonFunc\} \quad (2.12)$$

де *Func* – клас функціональних вимог, які можна представляти у вигляді шаблонів;

NonFunc – клас нефункціональних вимог до програмного забезпечення на другому рівні моделі зрілості.

Третій рівень моделі зрілості вимог до програмного забезпечення подано у нотації множини, елементами якої є:

- атрибути вимог;
- шаблони вимог;
- моделі вимог;
- контрольні листи;
- рекомендації.

Формально третій рівень моделі зрілості можна представити наступним чином:

$$L_3 = \{R_i^{use}, Patern_{ij}, RModel_{ik}, RList_{im}, Rcom_h\{L_2\{L_1\{L_0\}\}\}\} \quad (2.13)$$

де R_i^{use} – трансформовані у вимоги потреби замовника за допомогою першої моделі стандарту ISO 25010, $i=1..G$, G – кількість вимог;

$Patern_{ij}$ – патерни для опису вимог до ПЗ, $j=1..Q$, Q – кількість патернів;

$RModel_{ik}$ – моделі для опису вимог, $k=1..P$, P – кількість моделей вимог;

$RList_{im}$ – контрольні листи для перевірки вимог, $m=1..M$, M – кількість контрольних листів;

$Rcom_h$ - рекомендації щодо покращення вимог, h – кількість рекомендацій.

На четвертому рівні моделі зрілості будуються ієрархії вимог, встановлюються залежності між ними, визначаються типові рішення та ін.

Для формального представлення четвертого рівня моделі зрілості пропонується скористатись другою моделлю (зовнішня якість) стандарту [12]. Перехід від першої моделі до другої при встановленні відношень між вимогами пропонується здійснити за допомогою експертного оцінювання.

Формально четвертий рівень зрілості вимог представимо наступним чином

$$L_4 = \{R_i^{ext}, \{L_3\{L_2\{L_1\{L_0\}\}\}\}\} \quad (2.14)$$

П’ятий рівень зрілості вимог характеризується процесами трасування на компоненти проектування і тестування, визначення пріоритетів вимог, кількісного оцінювання вимог щодо трудозатрат та впровадженням засобів керування та інтеграції з інструментальними засобами імплементації ПЗ. Формально п’ятий рівень моделі зрілості можна представити

$$L_5 = \{R_i^{in}, \{L_4\{L_3\{L_2\{L_1\{L_0\}\}\}\}\}) \quad (2.15)$$

У даному випаду R_i^{in} інтерпретує модель для встановлення проміжних прототипів системи і відповідає третій моделі стандарту ISO 25010 (внутрішня якість).

Графічно модель з відповідними рівнями зрілості вимог і процесами наведено на рис. 2.7.

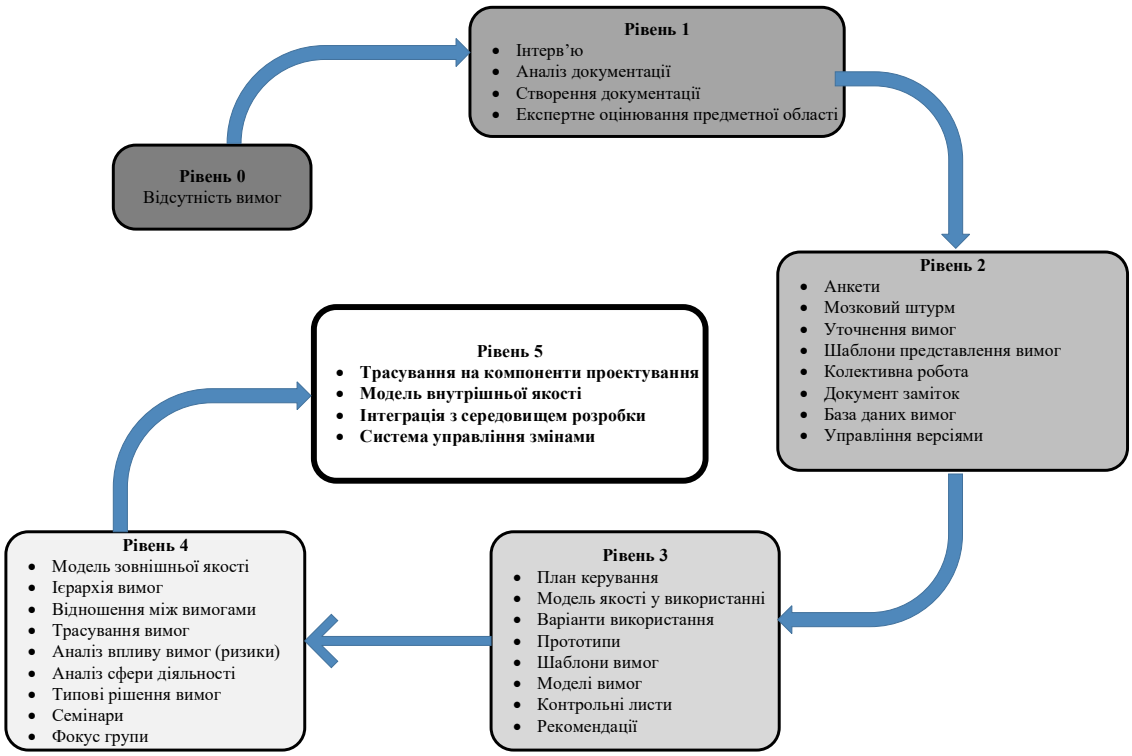


Рисунок 2.7 – Рівні зрілості вимог модифікованої моделі

Таким чином проведено формалізацію рівнів зрілості вимог до ПЗ, що дозволить більш ефективно впроваджувати управління вимогами та забезпечувати їх якість при низькому порозі входу для команди розробників та організації в цілому.

У даному розділі кваліфікаційної роботи одержано наступні результати:

1. Обґрунтовано модель зрілості RMM (Requirements maturity model) для управління зрілістю вимог до програмного забезпечення, що дало змогу підвищити якість процесу виявлення, аналізу та керування вимогами і забезпечити швидкий перехід команди-розробників з нульового на вищі рівні моделі зрілості.

2. Запропоновано модифікацію моделі RMM шляхом імплементації рекомендацій стандарту ISO 25010 на вищих рівнях моделі зрілості вимог, що дало змогу забезпечити адекватне їх відображення на наступних стадіях життєвого циклу та оптимізувати ресурси на їх реалізацію.

3. Формалізовано модифіковану модель зрілості вимог за допомогою апарату теорії множин та забезпечено вкладеність рівнів з безшовним переходом між ними, що дало змогу в подальшому запропонувати екосистему CASE-засобів та підвищити ефективність автоматизації процесів життєвого циклу ПЗ.

3 ПРОГРАМНА СИСТЕМА ЗБОРУ ПОТРЕБ ТА УПРАВЛІННЯ ЗРІЛІСТЮ ВИМОГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Аналіз функціональності сучасних систем управління вимогами

Сучасні інструментальні засоби спрощують роботу з вимогами, однак аналіз і відслідковування зв'язків між вимогами вимагає значних трудових ресурсів, особливо у випадках, коли кількість вимог перевищує декілька тисяч.

Аналіз сфери діяльності дозволяє оцінювати наявність залежності вимог високого рівня з вимогами більш низького рівня. Це дозволяє визначати, чи є у вимогах «непокриті» сценарії. Наприклад, якщо є ключова можливість, яка не зв'язана з жодним варіантом використання, то це в подальшому при реалізації програмної системи призведе до спрощення функціональності. І навпаки, якщо наявний сценарій використання, який не пов'язаний з жодною можливістю реалізації, то це може призвести до додаткової функціональності, яка не потрібна замовнику для вирішення його бізнес цілей. Аналіз сфери діяльності зручно проводити у вигляді матриць трасування і дерева трасування. Інструмент для трасування вимог IBM DOORS показано на рис. 3.1.

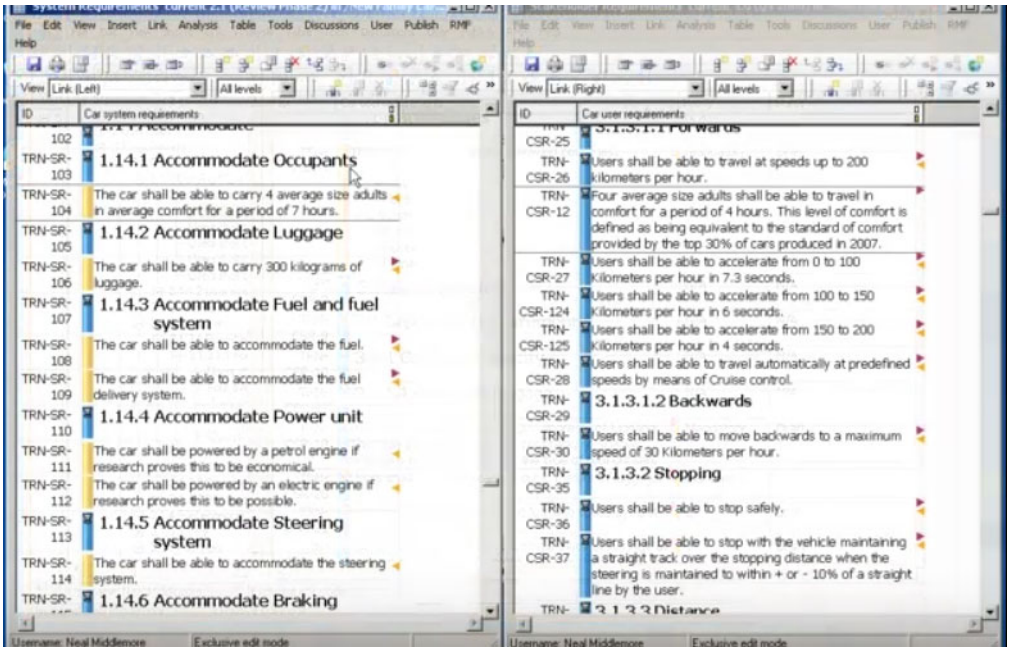


Рисунок 3.1 – Інструмент для трасування вимог IBM DOORS

Одним з підходів до документування вимог на четвертому рівні моделі зрілості є застосування підходу типових рішень. Типові рішення вимог є розширеною і покращеною версією вимог, що використовувались на третьому рівні моделі зрілості. Типові рішення є підходом подібним до підходу шаблонів і застосовуються для опису визначеного типу вимог певної функціональної області. Для прикладу, більшість систем опрацьовують одержану від користувача інформацію, тобто записують її в базу даних, читають або видаляють. Незалежно від типу даних система завжди буде виконувати ті сам дії. У такій ситуації для опису вимог можна застосувати підхід типового рішення.

На сучасному етапі розвитку комп'ютерної та програмної інженерії фахівцями в області інформаційних технологій застосовується велика кількість типових рішень [9, 12, 13]. Найбільш популярними серед них є типові рішення для опису вимог у вигляді варіантів використання.

Проектування комп'ютерних систем, які передбачають використання баз даних зазвичай використовує типове рішення CRUD («Create», «Read», «Update», «Delete»), що призначене для керування інформацією незалежно від джерел її надходження. Під керуванням інформацією в даному випадку розуміють запис, читання та оновлення даних у базі даних, а також видалення інформації.

Керування інформацією є потребою замовника або користувача системи, оскільки користувач буде працювати з інформацією. Спочатку здається, що було зручно створити чотири окремих варіанти використання: «Записати дані», «Оновити дані», «Прочитати дані», «Видалити дані». Однак кожен з наведених варіантів використання є атомарною операцією, яка виконується системою при реакції на дію користувача.

У випадку значної кількості різнорідної інформації, набір таких атомарних варіантів використання буде доволі великим, що призведе до ускладнення моделі і як наслідок до неможливості структурування вимог і керування ними. З точки зору практики, застосування наведених варіантів використання повинні бути реалізовані одночасно, оскільки відносяться до тієї ж інформації, що

опрацьовується. Для спрощення моделі варіантів використання і забезпечення її зрозумілості використовується типове рішення CRUD. У випадку використання CRUD буде створений один варіант використання «Управління інформацією». У випадку виконання додаткових дій при виконанні однієї з чотирьох операцій можна створити окремі сценарії одного варіанту використання.

Ще одним показником, який дозволяє говорити про задоволення потреб замовника у системі, є використання вимог на етапі тестування. Вимоги, в даному випадку, служать базисом для тестувальників при складанні тест плану, сценаріїв тестування та іншої необхідної інформації. При цьому для до загальних типів вимог необхідно додати такі типи вимог, які стосуються сценаріїв тестування, діаграм класів і діаграм взаємодії. Такі задачі повинні виконуватися шляхом застосування інструментальних засобів. Для прикладу, при використанні технології RUP в процесі розробки програмного забезпечення для управління вимогами можна використовувати Rational Request Pro, який в подальшому інтегрується з Rational Software Architect. Інтеграція цих засобів дає змогу відслідковувати зміни у вимогах на рівні проектування архітектури програмного забезпечення комп’ютерних систем.

Вимоги є базисом для процесу управління проектом, тому лідер команди та особа, відповідальна за проект, повинні володіти об’єктивними даними щодо стану проекту відносно реалізації вимог. Аналітики і керівники проекту можуть оцінювати і уникати ризиків, пов’язані з вимогами, шляхом додавання до кожної вимоги атрибут-ризик. У цьому атрибуті описують можливий ризик щодо вимоги і шляхи його уникнення та керування.

Кількісна оцінка вимог є пріоритетом при плануванні проекту. Вимоги за показниками кількості, складності реалізації, пріоритетами і ризиками є визначальними факторами в процесі планування проекту, зокрема формуванні бюджету і термінів виконання проекту. Найбільш складною задачею планування є визначення трудозатрат необхідних для опису, проектування і реалізації вимог, тобто кількісна оцінка кожної вимоги.

Для вирішення такої задачі можна використати підхід оцінювання на основі Use Case Points (UCP), як один із множини методів оцінювання трудозатрат в IT проектах. Даний підхід дозволяє аналітику оцінювати трудозатрати на опис варіантів використання і формувати пропозиції для лідера команди. Трудозатрати можуть бути додані у вигляді атрибутів вимог.

CASE-засоби дозволяють скоротити трудомісткість будь-якого процесу та забезпечити реалізацію принципів на яких базується технологія. На практиці в даний час використовується ряд технологій, серед яких найпопулярніші:

1. «Microsoft Solutions Framework».
2. «Custom Development Method Oracle».
3. «Rational Unified Process».

Проведемо аналіз цих технологій та відповідних засобів автоматизованої підтримки стадії проектування вимог.

3.1.1. Засоби автоматизації, що підтримують технологію Microsoft Solutions Framework

У [16] технологія Microsoft Solutions Framework означена як платформно-незалежна технологія, що орієнтована на розробку ПЗ і розвиток інформаційної інфраструктури. Також дану технологію можна визначити, виходячи з методів та засобів розробки, які вона використовує, тобто це комплекс засобів і методів процесу розробки ПЗ, що включає скоординований набір елементів (програмно–технічних засобів, документації, навчання и супроводу) [17]. Засоби цієї технології орієнтовані на розподілені обчислення та застосування технології «клієнт-серевер».

При проектуванні вимог до ПС Microsoft Solutions Framework базується на підході використання шаблонів та UML діаграм. Суть підходу, що використовує шаблони полягає у наданні можливості формалізації потреб до ПС та розділенні описової структури від даних при їх заповненні. В якості прикладу у розділі 1

наведено шаблон та одну із вимог (у вигляді шаблону), що визначає продуктивність web-сайту.

Недоліком технології Microsoft Solutions Framework є те, що використання шаблонів не є загальноприйнятим та стандартизованим, тобто кожна фірма-розробник може проектувати шаблони на свій розсуд. Крім того, для кожного нового проекту необхідно створювати нові шаблони, що не дає змоги повторного використання раніше створених. До переваг технології MSF можна віднести той факт, що специфікація вимог досить проста та зрозуміла, а також вирішено в більшій мірі питання уніфікованого представлення вимог.

Найбільш поширеними засобами автоматизованого проектування та фіксації вимог, що базується на технології MSF є Microsoft Visual Studio, Microsoft Visio та пакет Microsoft Office. Microsoft Visual Studio та Microsoft Visio підтримує генерацію UML-діаграм, зокрема в області проектування вимог використовує Use case діаграми. З допомогою Use case діаграм можна зобразити лише функціональні вимоги і неможливо описати нефункціональні вимоги, тобто вимоги якості. Документування вимог при використанні технології MSF здійснюється за допомогою текстового редактора MS Word, а розробка шаблонів виконується у MS Excel. Для автоматизованого керування та контролю загального процесу створення ПЗ використовується засіб Microsoft Project [18].

3.1.2. Автоматизовані системи виявлення вимог на основі технології Rational Unified Process (RUP)

Технологія RUP базується на принципах об’єктно-орієнтованого підходу створення ПС та мови графічного моделювання UML. Дана технологія підтримується інструментарієм групи Rational (IBM) [19].

Засобом, що підтримує моделювання діаграм цього типу є CASE-засіб Rational Rose. На основі змодельованих діаграм можна згенерувати вимоги до ПС

у вигляді спеціалізованої мови. Представлення вимог у вигляді спецмови для замовника є складним та незрозумілим.

Інший засіб керування вимогами та їх документування – середовище Rational Requsite Pro.

Основне призначення цього засобу зводиться до надання можливості відслідкування та зручного внесення змін у вимоги.

При цьому вимоги представляються у текстовому вигляді та базуються на відповідних діаграмах, що змодельовані в Rational Rose. Загалом перевагою технології RUP є те, що вона підтримується інструментарієм для автоматизації майже всіх етапів ЖЦ, які між собою тісно пов’язані та використовує ітераційні моделі створення ПЗ. Об’єктно-орієнтовний підхід при формулюванні вимог втілений також в автоматизованому засобі DOORS компанією Telelogic. Принцип проектування вимог відображає структуру шаблону, що рекомендований у стандарті [19].

3.1.3. Технологія CDM та інструмент її автоматизації

В основі технології Custom Development Method Oracle, згідно [20], лежить метод ORACLE CASE* METHOD. Даний метод базується на визначенні об’єктів (сутностей) і залежностей, що фактично є відображенням суті структурного підходу, в основу якого закладено використання діаграм «сутність-зв’язок». Технологія CDM підтримується інструментальними засобами компанії Oracle і використовується при розробці автоматизованих інформаційних систем на основі реляційних баз даних. До засобів автоматизації етапів життєвого циклу, що підтримують дану технологію, можна віднести наступні:

1. Oracle Designer;
2. Oracle Warehouse Builder;
3. JDeveloper;
4. Oracle9i Reports Developer.
5. Oracle Workflow.

При проектуванні та керуванні вимогами на практиці застосовується Oracle Designer. Даний засіб дозволяє моделювати діаграми «сутність-зв’язок», при цьому можна лише визначати основні сутності всередині системи та зв’язки між ними, але неможливо описати систему в цілому та процеси, які в ній протікають.

Структурний підхід забезпечують такі інструментальні засоби, як ARIS Toolset, Process Modeler (BPwin), ERwin Datamodeler та ін.

3.2. Проектування архітектури системи управління зрілістю вимог на вищих рівнях моделі RMM

Проаналізувавши існуючи підходи та засоби автоматизації процесом управління вимог запропоновано архітектуру програмної системи для управління зрілістю на вищих рівнях моделі RMM. Загальна архітектура ситсеми у вигляді діаграми компонентів програмного засобу на концептуальному рівні, яка представлена на рис. 3.2.

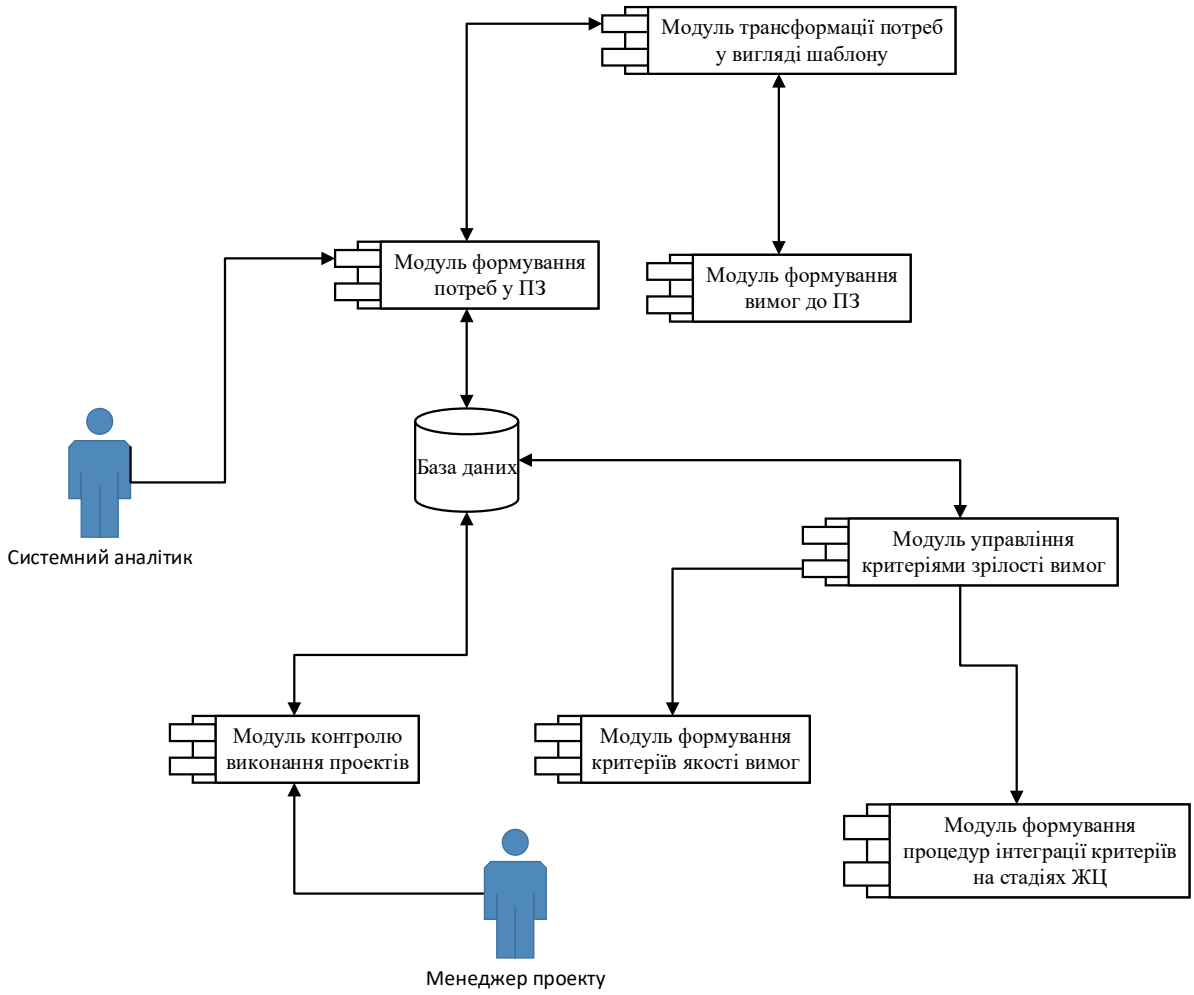


Рисунок 3.2 – Архітектура програмного засобу підтримки методу управління зрілістю вимог

Аналізуючи рис. 3.2 видно, що користувачами системи є два типи користувачів: менеджер проекту та системний аналітик. Архітектурні компоненти системи управління зрілістю вимог забезпечують виконання функцій щодо збору вимог, управління критеріями якості і спостереження за вимогами на стадіях життєвого циклу. Така організація системи дає змогу оптимізувати процеси імплементації програмного забезпечення. Проектування архітектури системи виконувалось із дотриманням правила, про те що на найвизому концептуальному рівні кількість компонентів системи знаходиться у межах 7±2 компоненти. Використання правил композицій програмних модулів забезпечується шляхом їх вкладеності.

База даних, структура якої наводиться у наступному підрозділі, зберігає інформацію щодо потреб замовника у ПЗ, детальних вимог до системи та

показників якості вимог.

3.3. Проектування схеми бази даних системи управління зрілістю вимог до ПЗ

У результаті аналізу предметної області та з врахуванням архітектурних аспектів при проектуванні програмної системи управління зрілістю вимог, спроектовано реалізовано засобами СКБД MS SQL Server реляційну базу даних, яка містить 17 таблиць.

На рис. 3.3 показано структуру таблиці «Project», що складається з трьох полів: первинний ключ, який забезпечує ідентифікацію проекту ПЗ, його назва та короткий опис.

ID_Project	int	☐
ProjectName	varchar(150)	☑
ProjectDescription	varchar(MAX)	☑

Рисунок 3.3 – Результат реалізації таблиці «Project»

Для зберігання даних про компоненти моделей якості стандарту ISO 25010 створено таблицю «Model» (рис. 3.4) до складу якої входить: ідентифікатор моделі, посилання зовнішнього ключа таблиці «Project», назва та короткий опис моделі.

Column Name	Data Type	Allow Nulls
ID_Model	int	☐
ID_Project	int	☑
ModelName	varchar(120)	☑
Description	varchar(MAX)	☑

Рисунок 3.4 – Результат реалізації таблиці «Model»

В якості окремої сутності визначено таблицю «Characteristic» (рис. 3.5),

основними полями якої є унікальне значення характеристики, її назва і зовнішній ключ, який вказує на приналежності до моделі.

	Column Name	Data Type	Allow Nulls
🔑	ID_Characteristic	int	☐
	ID_Model	int	☑
	CharacteristicName	varchar(150)	☑

Рисунок 3.5 – Результат реалізації таблиці «Characteristic»

Оскільки, характеристики моделей якості можуть містити підхарактеристики, то відповідно реалізовано таблицю «SubCharacteristic», структуру якої подано на рис. 3.6.

	Column Name	Data Type	Allow Nulls
🔑	ID_SubCharacteristic	int	☐
	ID_Characteristic	int	☑
	SubCharacteristicName	varchar(150)	☑
	Description	varchar(MAX)	☑

Рисунок 3.6 – Результат реалізації таблиці «SubCharacteristic»

Враховуючи наявність зв’язку «багато-до-багатьох» між таблицями «Characteristic» та «SubCharacteristic», виконано його декомпозицію, у результаті якої утворилось реляційне відношення, показане на рис. 3.7. Інтерпретація цього зв’язку пояснюється тим, що багато підхарактеристик можуть належати багатьом характеристикам якості.

	Column Name	Data Type	Allow Nulls
🔑	ID_Rel_Char_SubChar	int	☐
	ID_Characteristic	int	☑
	ID_SubCharacteristic	int	☑

Рисунок 3.7 – Відношення «Rel_Char_SubChar»

Представлення вимоги до ПЗ у вигляді окремого атрибуту демонструє

таблиця «Attribute», структуру якої показано на рис. 3.8. Окрім унікального ідентифікатора до її складу входять назва і призначення атрибуту, а також посилання на таблицю»Rel_Char_SubChar».

	Column Name	Data Type	Allow Nulls
🔑	ID_Attribute	int	☐
	ID_Rel_Char_SubChar	int	☑
	AttributeName	varchar(150)	☑
	Description	varchar(MAX)	☑

Рисунок 3.8 – Результат реалізації таблиці «Attribute»

Кількісну міру кожної вимоги у вигляді атрибуту одержують за допомогою відповідної метрики, що передбачає створення одноіменної таблиці, представленої на рис. 3.9.

	Column Name	Data Type	Allow Nulls
🔑	ID_Metric	int	☐
	MetricName	varchar(150)	☑
	Formula	varchar(100)	☑
	Description	varchar(MAX)	☑

Рисунок 3.9 – Реляційне відношення «Metric»

По аналогії до характеристик і підхарактеристик, між вимоогою-атрибутом та її метрикою існує залежність «багато-до-багатох», тому релаізовано проміжну таблицю для уникнення такої аномалії під назвою «Attr_Metric» (рис. 3.10)

	Column Name	Data Type	Allow Nulls
🔑	ID_Attr_Metric	int	☐
	ID_Attribute	int	☑
	ID_Metric	int	☑

Рис. 3.10. Attr_Metric

Потреби замовника ПЗ представляються у вигляді таблиці «Needs», яка

показана на рис. 3.11.

	Column Name	Data Type	Allow Nulls
🔑	ID_Need	int	☐
	NeedNumber	int	☑
	NeedDescription	varchar(MAX)	☑

Рисунок 3.11 – Таблиця «Needs»

Для вказання відображення потреб у вимоги, тобто встановлення зв’язку між таблицями «Attribute» та «Needs» реалізовано реляційне відношення «Need_Attribute»(рис. 3.12).

	Column Name	Data Type	Allow Nulls
🔑	ID_Need_Attribute	int	☐
	ID_Attribute	int	☑
	ID_Need	int	☑

Рисунок 3.12 – Need_Attribute

Окрім цього у базі даних реалізовано інші таблиці, які мають відношення до вимог на інших етапах життєвого циклу, методології Agile при виконанні програмних проектів, а також задач і розробників, які їх реалізовують. В загальному випадку структура бази даних (ER-діаграма) представлена на рис. 3.13.

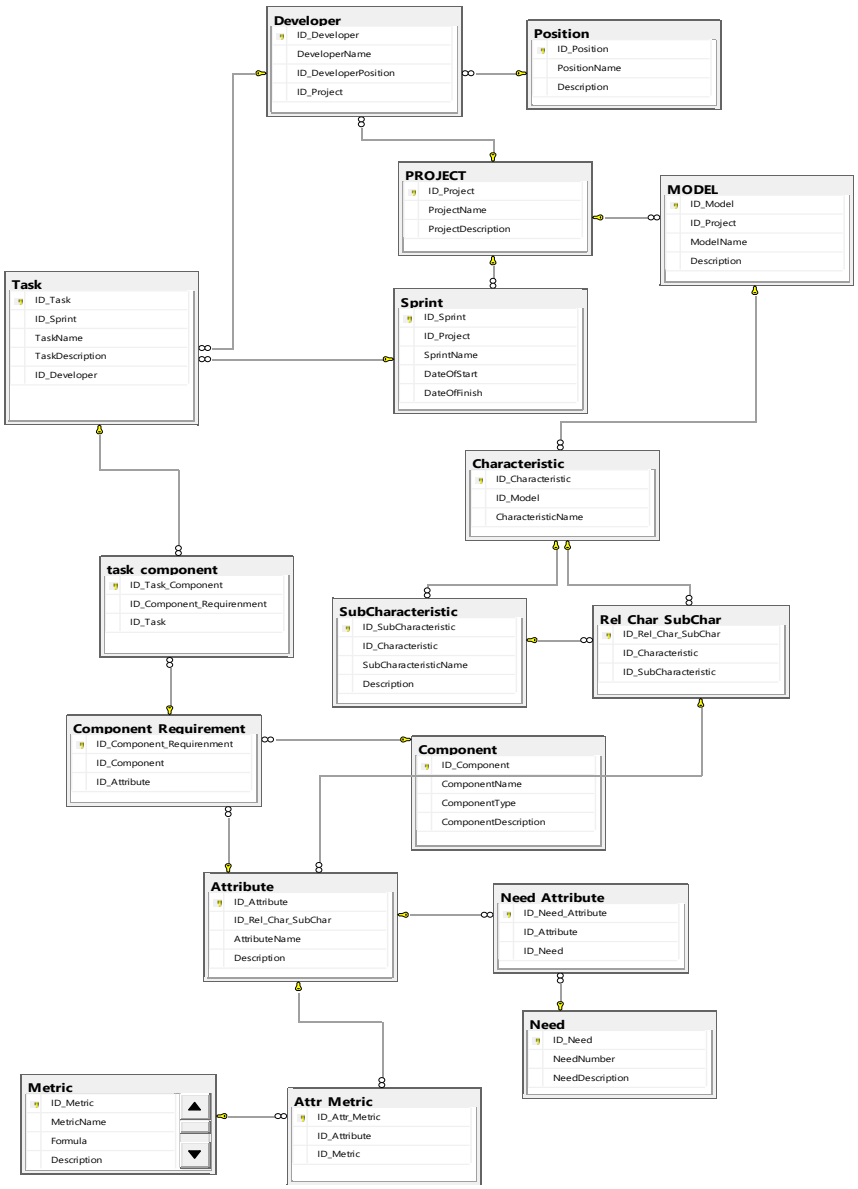


Рисунок 3.13 – ER-діаграма бази даних

3.4. Реалізація інтерфейсів системи управління зрілістю вимог

Програмну реалізацію системи управління зрілістю вимог реалізовано засобами мови PHP і стеку front end технологій JavaScript, HTML/CSS.

Формулювання потреб замовника може відбуватись у двох режимах: на основі шаблону та у формі звичайного неформалізованого тексту. Вибір режиму задання потреб замовник може здійснити після успішної ідентифікації за допомогою форми, наведеної на рис. 3.14.

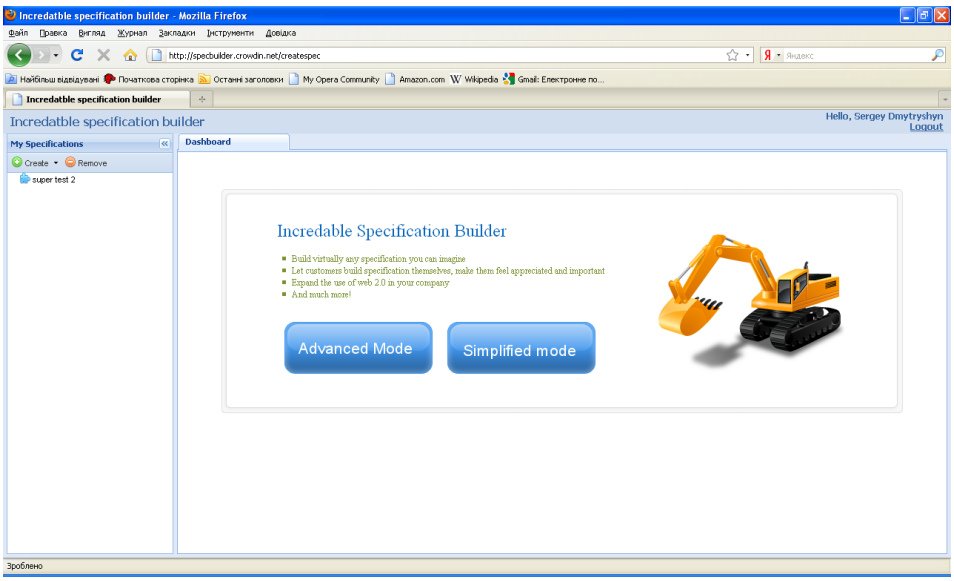


Рисунок 3.14 – Інтерфейс для формування вимог у вигляді структури моделей якості

Для системного аналітика передбачено патерн, який дозволяє на основі потреб замовника сформулювати вимогу у вигляді структурних компонентів моделі якості. При цьому, аналітик і замовник має можливість вибору атрибутів, які потрібно реалізувати в першу чергу (рис. 3.15).

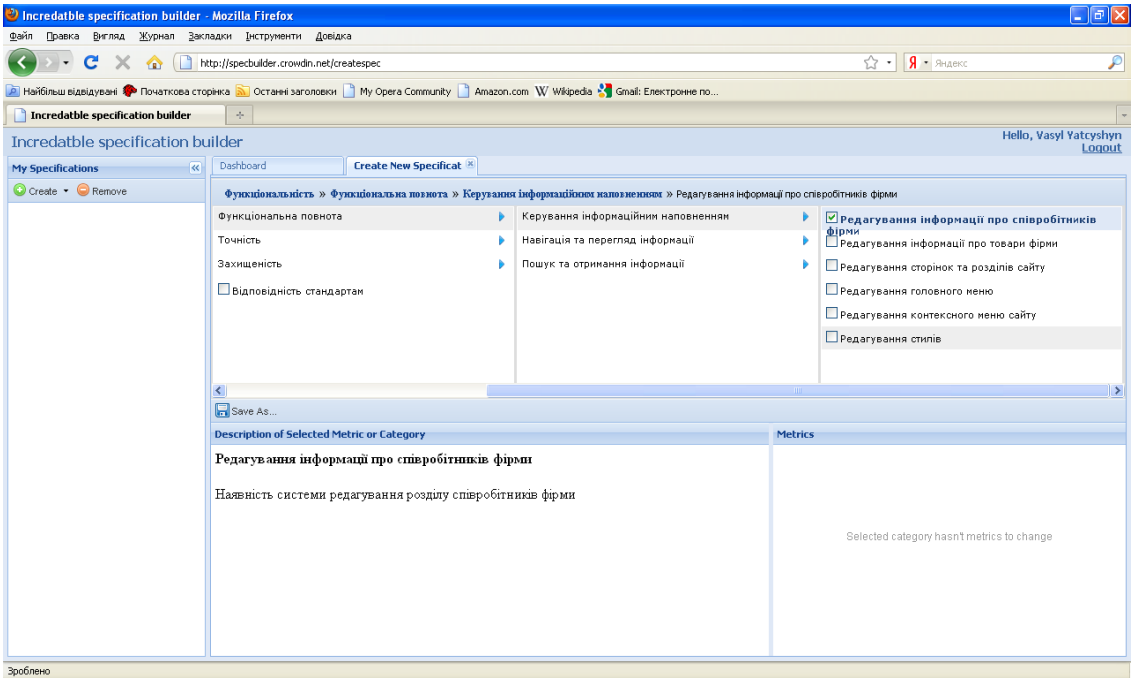


Рисунок 3.15 – Патерн представлення вимоги у вигляді структурних елементів моделі якості

Окрім цього, у системі управління зрілістю вимог наявна можливість створення потреб у простому текстовому представленні (рис. 3.16), після чого може бути виконана їх трансформація у вигляді патерну (рис. 3.15).

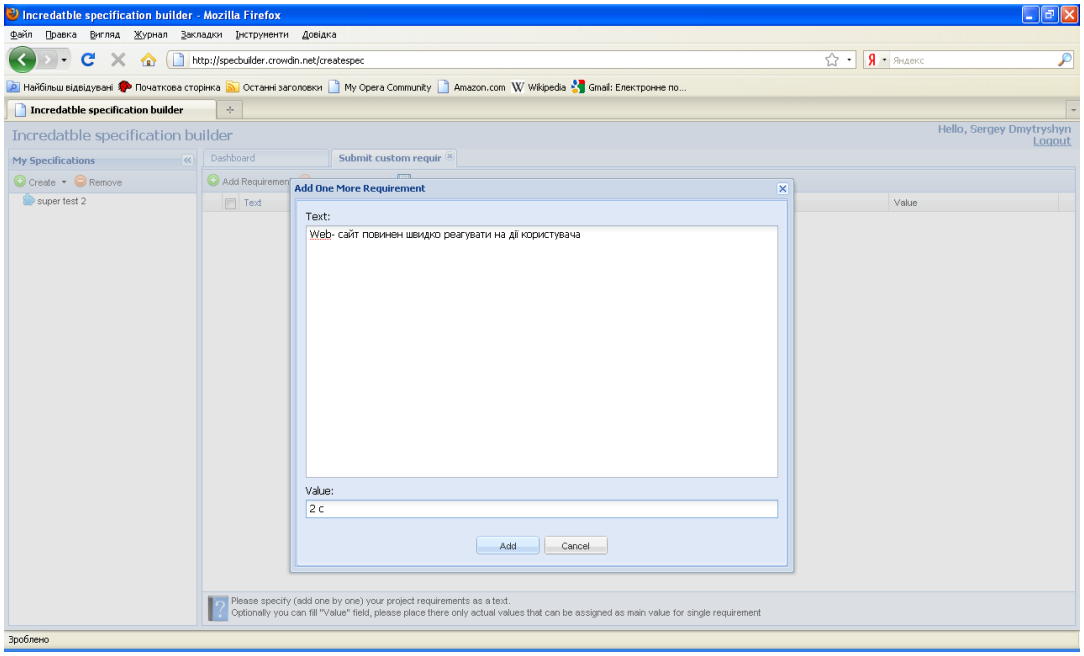


Рисунок 3.16 – Інтерфейс простого текстового формування користувацьких потреб

Backend системи управління зрілістю вимог до ПЗ імплементує можливості менеджера для взаємодії зі сховищем вимог і потреб, а також інших необхідних показників. Зовнішній вигляд Backend частини наведено на рис. 3.17.

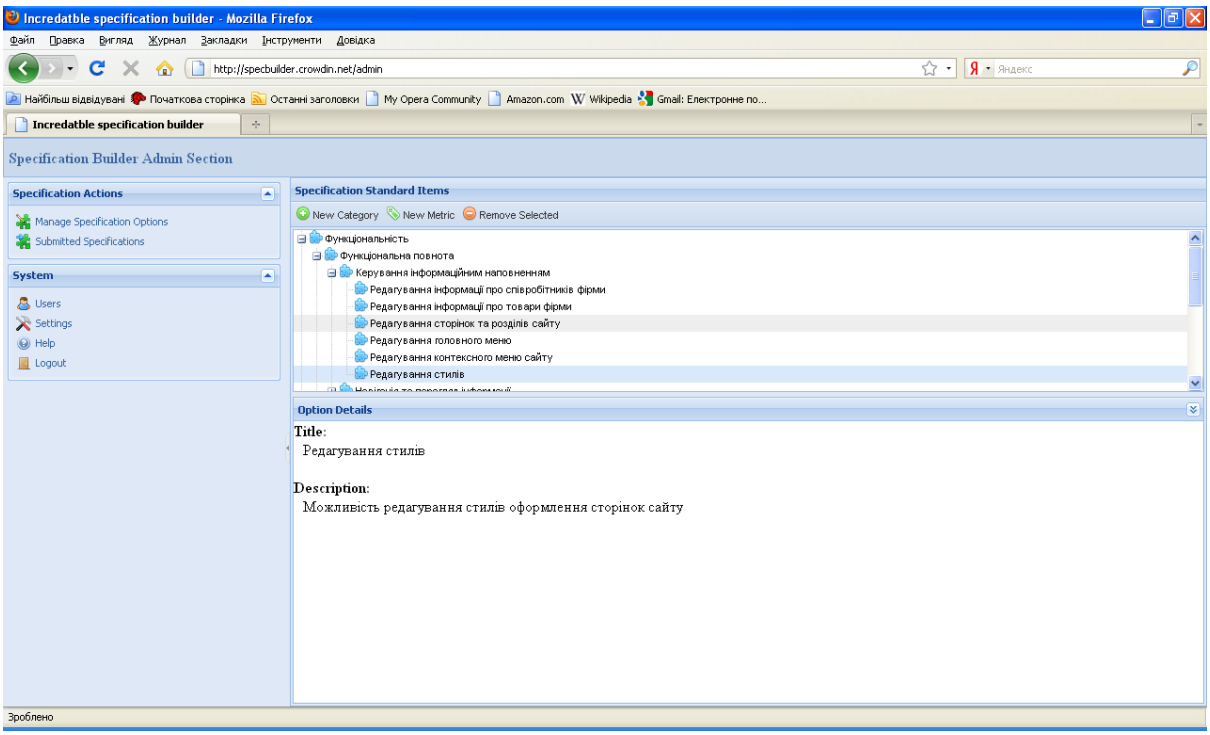


Рисунок 3.17 – Форма редагування

Для адміністратора системи управління зрілістю вимог передбачено функції моніторингу, управління та наповнення сховищ конкретними екземплярами компонентів моделей якості. Окрім цього, у його розпорядженні наявні всі інструменти, притаманні будь-якому класичному адміністратору системи, зокрема, можливість налаштування прав доступу до системи, зовнішніх користувацьких інтерфейсів і т.п.

Однак, з розвитком та інтенсивним впровадження інтернет технологій в процес розробки ПЗ, необхідно забезпечувати повноту потреб замовника у ПЗ. Зараз при комунікації команди розробників та замовника використовується різні засоби віддаленого спілкування (месенджери): Skype, Viber, Slack і багато інших, а також електронна пошта. Тому актуальною задачею є розробка рішення для збору потреб замовника і зацікавлених сторін з обґрунтованим засобом управління вимогами на основі моделей якості.

3.5. Створення модуля збирання вимог із джерел зовнішньої комунікації

Для роботи зі збору вимог до програмного забезпечення при проектуванні комп'ютерних систем та подальшої інтеграції з обґрунтованим CASE-засобом управління зрілістю вимог запропоновано скористатись платформою Onlizer.

Для того, щоб розпочати роботу із системою Onlizer необхідно провести реєстрацію на сайті http://portal.onlizer.com/, після чого увійти у особистий кабінет використовуючи форму для входу. Якщо авторизація пройшла успішно, то буде відкрита головна сторінка системи(рис. 3.18), на якій можна побачити існуючі Application, а також які Workflows є в цьому Application та їхній статус.

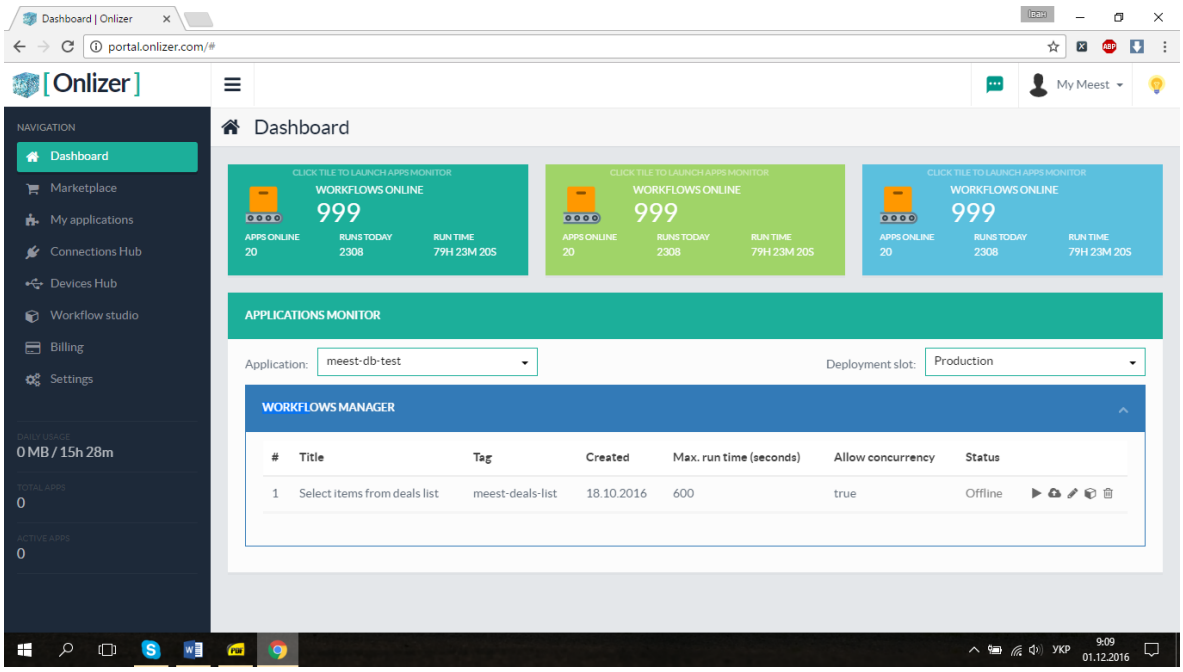


Рисунок 3.18 – Головна сторінка системи Onlizer

Щоб це зробити, спочатку потрібно створити application та workflow, відповідно до кроків описаних вище. Коли вони будуть створені у Workflow studio, на панелі справа потрібно буде знайти конектор HTTP Endpoint, конектори до Skype, Viber, Telegram та Slack, і відповідно конектор для зберігання інформації (потреб користувача) у базу даних MS SQL Server.

Коли всі компоненти будуть з’єднані можна приступити до їхніх налаштувань, щоб це зробити потрібно в контекстному меню конектора обрати пункт Settings. Під час першої настройки потрібно буде обрати метод, який буде використовувати конектор, після чого відкриється вікно налаштувань(рис. 3.19) де потрібно буде ввести відповідні параметри методу.

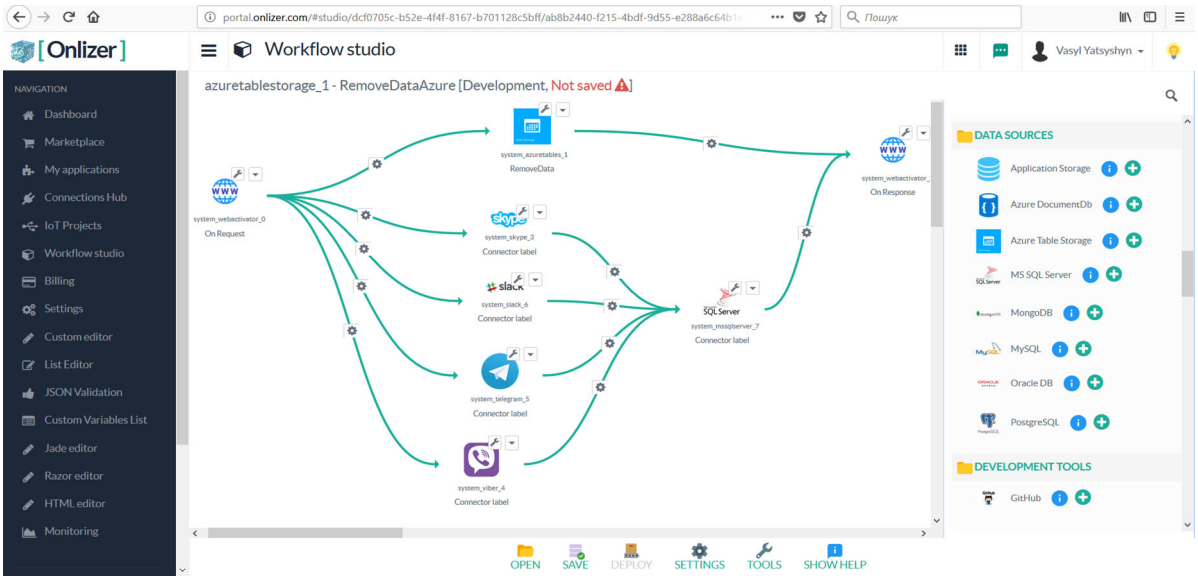


Рисунок 3.19 – Workflow для збору потреб замовника

Коли всі компоненти будуть налаштовані потрібно натиснути кнопку Save для збереження workflow. Збережений workflow можна запустити натиснувши кнопку Start у вікні настройок application.

У результаті одержимо записи у БД про потреби конкретної зацікавленої особи у програмному забезпеченні, у вигляді структури полів відповідної таблиці. Полями таблиці є: ідентифікатор запису; тип засобу комунікації; час комунікації; контактна особа зі сторони замовника; контактна особа зі сторони розробника; вміст повідомлення (або посилання на файл); час запису.

В такому випадку, інтеграція розробленого модуля збору вимог та CASE засобу управління вимогами на основі моделі якості здійснюється через спільний доступ до БД потреб.

Архітектура засобу підтримки моделі зрілості вимог програмного забезпечення матиме вигляд, як показано на рис. 3.20.

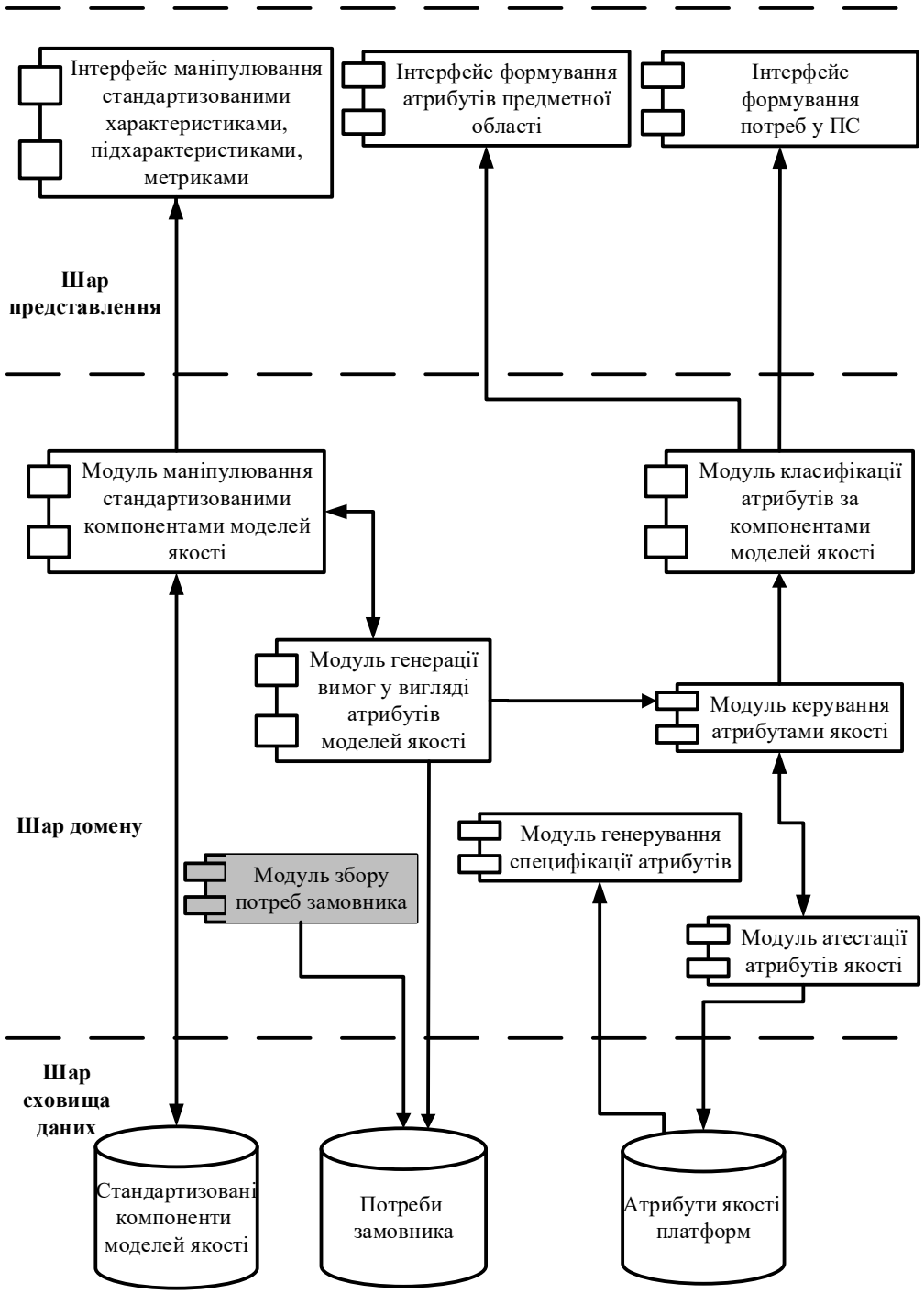


Рисунок 3.20 – Архітектура засобу підтримки рівнів зрілості модифікованої моделі RMM

Розроблений модуль збору потреб замовника із додаткових джерел таких як Skype, Viber, Slack, Telegram дає можливість доповнювати потреби замовника у

програмному забезпеченні та в комп’ютерній системі в цілому. За рахунок цього модуля забезпечується більша повнота вимог до програмного забезпечення, яке необхідно реалізувати.

Інтеграція розробленого модуля у CASE засіб управління та розробки вимог на основі моделей якості здійснюється на основі зв’язку з базою даних потреб замовника, що знаходиться у шарі сховища даних. Даний модуль є повністю ізольований від CASE засобу і працює як сервіс, який можна запускати за розкладом при встановленні комунікації між командою розробників і замовником системи. Оплата за використання модуля здійснюється лише за реальний час його використання. Коли модуль не включений, оплата відповідно не проводиться.

Таким чином, у дипломній роботі магістра запропоновано модель покращення якості процесу розробки вимог до програмного забезпечення та розроблено інструментарій для підтримки моделі зрілості вимог.

У даному розділі кваліфікаційної роботи магістра одержано наступні практичні результати:

1. Проведено аналіз сучасних інструментів підтримки процесів життєвого циклу ПЗ у результаті якого виявлено їхні недоліки та здатність до взаємодії з іншими засобами автоматизації, що дало можливість виявити потенційні шляхи впровадження модифікованої моделі зрілості вимог.

2. Спроектовано архітектуру програмної системи управління зрілістю вимог, що відображає процес розробки програмного забезпечення із застосуванням методології Agile та інтегрованого представлення моделей якості, що дало змогу в повній мірі забезпечити автоматизацію управління зрілістю вимог на основі модифікованої моделі RMM.

3. Розширено можливості щодо збору потреб замовників з сучасних каналів комунікації шляхом використання інтеграційної платформи Onlizer, що дало змогу підвищити рівень повноти вимог та автоматизувати початкові рівні управління зрілості вимог.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

Кваліфікаційна робота магістра присвячена дослідженню і розробці методів і системи управління зрілістю вимог на етапах життєвого циклу програмного забезпечення. Оскільки, розроблена за допомогою мови програмування PHP система, реалізовувалась з використанням комп'ютерної техніки і розгорнута на веб-сервері в мережі Інтернет, то важливим було дотримання вимог і правил з охорони праці і техніки безпеки.

У загальному випадку, поняття охорони праці в галузі інформаційних технологій, представляє собою дотримання всіх вимог і нормативів, що присутні в законодавчих актах про охорону праці. Закони цієї області спрямовані на якісну і безпечну експлуатацію апаратних пристроїв і приміщень, дотримання санітарно-гігієнічних умов праці і захист від інших небезпечних чинників на підприємстві [24]. Ці засоби є складовими дослідження методів і системи управління зрілістю вимог при виконанні процесів життєвого циклу. В основних законодавчих актах про охорону праці приділяється велика увага поліпшенню умов праці в усіх галузях господарства, впровадженню сучасних засобів техніки безпеки і забезпечення санітарно - гігієнічних умов, що запобігають виробничому травматизму і професійним захворюванням.

Охорона життя і здоров'я людини є пріоритетним напрямком соціальної політики держави. В Україні прийнято закон прямої дії «Про охорону праці», який регламентує захист конституційного права працівників на безпечні умови праці. Законодавство України про охорону праці складається із загальних законів України та спеціальних законодавчих актів [24]. Загальними законами України, що визначають основні положення з охорони праці є Конституція України, Закон України «Про охорону праці», Кодекс законів про працю (КЗпП), Закон України «Про загальнообов'язкове державне соціальне страхування від нещасного випадку

на виробництві та професійного захворювання, які спричинили втрату працездатності» [24].

При розробці програмної системи управління зрілістю вимог на стадіях життєвого циклу ПЗ, які передбачали використання ПК, площа та об'єм для одного робочого місця оператора визначається згідно норм НПАОП 0.00-7.15-18, зокрема площа повинна становити не менше 6,0 квадратних метрів, об'єм - не менше 20,0 кубічних метрів [25].

Згідно вимог НПАОП 0.00-7.15-18 стіни, стеля та підлога приміщень, в яких розміщені комп’ютери, повинні бути виготовлені з матеріалів, дозволених для оформлення приміщень органами державного санітарно-епідеміологічного нагляду.

Заземлені конструкції, що знаходяться в приміщеннях, де розміщені робочі місця операторів (батареї опалення, водопровідні труби, кабелі із заземленим відкритим екраном), повинні бути надійно захищені діелектричними щитками та сітками з метою недопущення потрапляння працівника під напругу.

Організація робочого місця оператора повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам НПАОП 0.00-7.15-18.

У приміщенні, де одночасно експлуатуються понад п'ять електронно-обчислювальних машин, на помітному та доступному місці мають бути встановленні аварійні резервні вимикачі, які можуть повністю вимкнути електричне живлення приміщення, крім освітлення [26].

Одним із найбільш важливих нормативних документів щодо забезпечення охорони праці користувачів ПК є "Державні санітарні норми і правила роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин" ДСанПіН 3.3.2.007-98 [25].

Дотримання даних правил значно знижує наслідки несприятливої дії на працівників шкідливих та небезпечних факторів, які супроводжують роботу з відео-дисплейними матеріалами, зокрема можливість зорових, нервово-емоційних переживань, серцево-судинних захворювань. Виходячи з цього, роботодавець

повинен забезпечити гігієнічні й ергономічні вимоги щодо організації робочих приміщень для експлуатації електронно-обчислювальних машин (ЕОМ) з ВДТ, робочого середовища, робочих місць з ЕОМ, режиму праці і відпочинку при роботі з ЕОМ тощо, які викладені у нормах НПАОП 0.00-7.15-18.

Відповідно до встановлених гігієнічно-санітарних вимог роботодавець зобов'язаний забезпечити в приміщеннях з оптимальні параметри виробничого середовища [25].

Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски з поверхні екранів персонального комп’ютера і клавіатури повинні бути передбачені сонцезахисні пристрої, вікна повинні мати жалюзі або штори.

Вимогам нормативних актів з охорони праці мають відповідати:

– умови праці на кожному робочому місці;

– безпека технологічних процесів, машин, механізмів, обладнання й інших засобів виробництва;

– стан засобів колективного та індивідуального захисту;

– санітарно-побутові умови.

Приміщення з ЕОМ мають бути оснащені переносними вуглекислотними вогнегасниками з розрахунку 1 на 50 м^2, але не менше 2 на приміщення. Підходи до засобів пожежогасіння повинні бути вільними [25].

При облаштуванні робочих місць необхідно забезпечувати належні умови освітлення приміщення і робочого місця, оптимальні параметри мікроклімату, ергономічних характеристик основних елементів робочого місця, а також враховувати наявність шуму і вібрації, м'якого рентгенівського випромінювання, електромагнітного випромінювання, ультрафіолетового та інфрачервоного випромінювання, електростатичного поля між екраном і оператором, відсутність пилюки, озону, оксидів азоту та аероіонізації.

Отже, при розробці програмної системи управління зрілістю вимог на стадіях життєвого циклу ПЗ, проаналізовано та враховано необхідні норми щодо охорони праці при використанні електронно-обчислювальної техніки і забезпечено умови для зручної та ефективної роботи працівників.

4.2. Фактори, що впливають на функціональний стан користувачів комп’ютера

Робота відеотерміналів включає різні завдання, які об'єднуються такими загальними чинниками, як те, що робота проводиться в сидячому положенні і вимагає уважного, неперервного та іноді тривалого спостереження [26].

Виділяють три групи основних завдань, які розв'язуються на відеотерміналах:

– контроль і спостереження;

– діалог;

– збір інформації.

Ці завдання розрізняються по тривалості використання дисплея і по ступеню уваги, якого вони вимагають. Важливим питанням є режим праці і відпочинку при роботі з відеотерміналами. Виділяють 7 умов для того, щоб діяльність на робочому місці, оснащеному дисплеєм, здійснювалася без скарг і без втоми [26].

Правильне облаштування робочого столу:

– при фіксованій висоті – оптимальна висота - 720мм;

– повинен забезпечуватися необхідний простір для рук по висоті, ширині і глибині;

– в області сидіння не повинно бути шухляд.

Правильне встановлення робочого стільця:

– висота повинна регулюватися;

– конструкція повинна бути такою, що обертається;

– правильна висота сидіння: площа сидіння на 30мм нижче, ніж підколінна западина.

Правильне розташування приладів: необхідно так установити яскравість знаків і яскравість фону дисплея, щоб не було великої відмінності в порівнянні з яскравістю навколишнього оточення, але щоб знаки чітко пізнавалися на відстані читання. Не допускати, згідно [26]:

– дуже велику яскравість (викликає мерехтіння);

– дуже слабку яскравість (сильне навантаження на очі);

– дуже чорну фонову яскравість дисплея (сильне навантаження а очі).

Правильне виконання робіт:

– положення тулуба пряме, ненапружене;

– положення голови пряме, вільне, зручне;

– положення рук - зігнуті трохи більше, ніж під прямим кутом;

– положення ніг - зігнуті трохи більше, ніж під прямим кутом;

– правильна відстань для зору, клавіатура і дисплей –приблизно на однаковій відстані для зору: при постійній роботі - близько 500мм, при випадковій роботі - до 700мм.

Правильне освітлення:

– освітлення по можливості із сторони, зліва;

– по можливості - рівномірне освітлення всього робочого простору;

– прилади по можливості встановлювати в місцях, віддалених від вікон;

– вибирати непряме освітлення приміщення або вкривати корпуси світильників;

– світло, що поступає через вікна, пом'якшувати за допомогою штор;

– організувати робоче місце, щоб напрям погляду йшов по можливості паралельно фронту вікон.

Правильне застосування допоміжних засобів: підлокітники використовувати, якщо клавіатура вища 15мм [26].

Правильний метод роботи:

– передбачати по можливості зміну завдань і навантажень;

– дотримувати перерви в роботі: 5 хвилин через 1 годину роботи біля дисплея або 10 хвилин після 2-х годин роботи біля дисплея.

При створенні сприятливих умов для підвищення продуктивності і зменшення напруги значну роль грають чинники, що характеризують стан навколишнього середовища: мікроклімат приміщення, рівень шуму і освітлення. рекомендована величина відносної вологості - 65-70%. робоче місце повинне добре вентилюватися. В даний час з погляду шумового навантаження досягнутий значний прогрес. Рівень шуму в залі (приблизно 40дб) не перевищує рівень КБ, незалежно від кількості використовуваної апаратури. По останніх дослідженнях - робота за відеотерміналом не представляє небезпеки з погляду рентгенівського випромінювання [25].

Ергономічна організація робочого місця користувача ЕОМ повинна враховувати як специфіку діяльності, що виконується, так забезпечувати комфортні умови перебування людини.

Тому до основних ергономічних завдань щодо організації робочого місця слід віднести [25]:

– забезпечення просторових параметрів робочого місця, які відповідають антропометричним характеристикам користувача;

– раціональне розташування елементів робочого місця відносно користувача на підставі поглибленого кількісного та якісного аналізу діяльності, яка виконується;

– оптимізацію умов робочого середовища.

В ході організації робочих місць на кожну ЕОМ повинна бути виділена площа, яка складає не менш, ніж 6 м^2, та об'єм, який становить не менш, ніж 24 м^3. Причому, зона, де розташовується робочий стіл, сервер або робоча станція, принтер, екран для графопроектора, повинна займати відповідно 6 - 8 м^2. Висота приміщення повинна бути не менш, ніж 4 м [26].

Робоче місце користувача ПК повинно бути обладнане одномісним столом та напівм'яким стільцем, висоту сидіння яких можна змінювати. Довжина стола повинна бути не менше 70 см, ширина – забезпечувати місце перед клавіатурою (не менше, ніж 40 см) для розташування зошита або іншого приладдя. Поверхня

стола повинна мати кут нахилу у межах 12-150, лише іноді припустимою є її розташування у горизонтальній площині.

Слід відмітити, що оптимальна відстань від очей до площини екрана монітора, повинна складати 60–70 см, припустима – не менше 50 см. Розглядати інформацію на екрані з відстані менш, ніж 50 см не рекомендується.

ВИСНОВКИ

У кваліфікаційній роботі магістра одержано наступні наукові і практичні результати.

1. На основі аналізу основних понять та класифікації вимог до програмного забезпечення визначено особливості процесів інженерії вимог та обгрунтовано актуальність дослідження моделей, методів і засобів управління зрілістю на стадіях життєвого циклу, що в перспективі забезпечить можливість їх оптимізації шляхом підвищення ефективності як самих процесів, так і якості готового програмного продукту.

2. Проведено аналітичний огляд методів та інструментів інженерії вимог у результаті якого визначено критерії якості при формулюванні вимог та підходи до реалізація цього процесу і встановлено, що на сьогодні найбільш ефективними є застосування підходів, які використовуються структуру специфікації рекомендовану стандартом IEEE 830, технологію використання шаблонів на основі структурного та об’єктно-орієнтованого підходів, а також технологію моделей якості.

3. Створено формалізовану схему щодо оцінювання підходів до формулювання вимог ПЗ, яка використовує експертні оцінки і дає змогу кількісно представити ефективність їхнього застосування в задачах забезпечення та управління зрілістю вимог на стадіях життєвого циклу. Встановлено, що кращим варіантом є використання моделей якості для представлення вимог до ПЗ.

4. Обґрунтовано модель зрілості RMM (Requirements maturity model) для управління зрілістю вимог до програмного забезпечення, що дало змогу підвищити якість процесу виявлення, аналізу та керування вимогами і забезпечити швидкий перехід команди-розробників з нульового на вищі рівні моделі зрілості.

5. Запропоновано модифікацію моделі RMM шляхом імплементації рекомендацій стандарту ISO 25010 на вищих рівнях моделі зрілості вимог, що дало змогу забезпечити адекватне їх відображення на наступних стадіях життєвого циклу та оптимізувати ресурси на їх реалізацію.

6. Формалізовано модифіковану модель зрілості вимог за допомогою апарату теорії множин та забезпечено вкладеність рівнів з безшовним переходом між ними, що дало змогу в подальшому запропонувати екосистему CASE-засобів та підвищити ефективність автоматизації процесів життєвого циклу ПЗ.

7. Проведено аналіз сучасних інструментів підтримки процесів життєвого циклу ПЗ у результаті якого виявлено їхні недоліки та здатність до взаємодії з іншими засобами автоматизації, що дало можливість виявити потенційні шляхи впровадження модифікованої моделі зрілості вимог.

8. Спроектовано архітектуру програмної системи управління зрілістю вимог, що відображає процес розробки програмного забезпечення із застосуванням методології Agile та інтегрованого представлення моделей якості, що дало змогу в повній мірі забезпечити автоматизацію управління зрілістю вимог на основі модифікованої моделі RMM.

9. Розширено можливості щодо збору потреб замовників з сучасних каналів комунікації шляхом використання інтеграційної платформи Onlizer, що дало змогу підвищити рівень повноти вимог та автоматизувати початкові рівні управління зрілості вимог.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Астелс Д., Миллер Г., Новак М. Практическое руководство по экстремальному программированию. М.: «Вильямс». 2012. 320с.

2. ДеМарко Т., Листер Т. Вальсируя с медведями: управление рисками в проектах по разработке программного обеспечения. Компания p.m. Office. М. 2015. 190 с.

3. Лавріщева К.М. Програмна інженерія. К. 2008. 312 с.

4. Лаврищева Е.М. Методы программирования. Теория, инженерия, практика.К.: Наук. Думка. 2006. 451 с.

5. Брауде Е. Технология разработки программного обеспечения. СПб. : Изд-во "Питер" .2004. 655 с.

6. Вершина А. Модель процесса разработки программного обеспечения. Пробл. програмув. № 2-3 [спец. вип.]. 2006. С. 269-274.

7. Вигерс К. Разработка требований к программному обеспечению. М.: Издательско-торговый дом «Русская Редакция».2004. 576с.

8. ДСТУ 2844 -94. Програмні засоби ЕОМ. Забезпечення якості. Терміни та визначення. Чинний від 01.01.96. К. Держстандарт України. 1995. 15 с.

9. ДСТУ 2853 -94. Програмні засоби ЕОМ. Підготовлення та проведення випробувань. Чинний від 01.01.96. К. Держстандарт України.1994. 17 с.

10. ДСТУ 3918-1999 (ISO/IEC 12207:1995) Інформаційні технології. Процеси життєвого циклу програмного забезпечення, К.: Держстандарт України. 2000. 49 с.

11. Ларман К. Применение UML и шаблонов проектирования М.: Вильямс, 2012. 496 с.

12. Лаврищева Е. Методы и средства инженерии программного обеспечения: Учебник. М.: МФТИ (ГУ). 2006. 304 с.

13. Маторин С. Анализ и моделирование бизнес-систем: системологическая объектно-ориентированная технология. Харьков: ХНУРЭ. 2002. 322 с.

14. Харченко О. CASE-технологія розроблення вимог до програмного продукту, та оцінювання його якості. Науковий вісник НЛТУ України. Вип.20.2. 2010. С. 277 –285.

15. Фаулер М. Архитектура корпоративных программных приложений /М. Фаулер. М.: Издательский дом "Вильямс". 2006. 544 с.

16. Capability Maturity Model / M. Paulk, B. Curtis, M. Chrissis, Ch. Weber – IEEE Software. Vol. 10. 4. 1993. P. 18–27.

17. Фатрелл Р., Шафер Д., Шафер Л. Управление программными проектами: достижение оптимального качества при минимуме затрат. М.: Издательский дом «Вильямс». 2013. 1136 с.

18. Брауде Э. Технология разработки программного обеспечения. СПб: Питер, 2014. 655 с.

19. ISO/IEC 12207:2008. Systems and software engineering – Software life cycle processes

20. ISO/IEC 33001:2015. Information technology – Process assessment

21. Кантор М. Управление программными проектами. Практическое руководство по разработке успешного программного обеспечения. М.: Вильямс. 2012. 176 с.

22. Britkin A.I. Model estimate the duration of the iterative software development process. Open education. 2009. №4. P. 75.

23. Sommerville I. Software Engineering. Addison-Wesley Publ. Company. 2011. 866 p.

24. Желібо Є., Заверуха Н., Зацарний В. Безпека життєдіяльності. К.: 2001. 483 с.

25. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями». Київ. 2018. URL: http://sop.zp.ua/norm_npaop_0_00-7_15-18_01_ua.php (дата звернення: 10.11.2021).

26. Бедрій Я. Основи охорони праці користувачів персональних комп’ютерів: навчальний посібник для студентів ВНЗ та інженерів-практиків. Навчальна книга-Богдан. 2014. 144 с.

ДОДАТОК А

Апробація результатів роботи

ДОДАТОК Б

Фрагменти коду програмної системи управління зрілістю вимог

Фрагмент коду Б.1. Лістинг частини контролера при реалізації серверної частини

```
<?php
class AdminController extends Zend_Controller_Action
{
    public function init() {
      if(!isset($_SESSION['admin'])) {
        header("Location: /admin-login");
        die();
      }
    }
    public function indexAction() {
    }
    public function usersAction() {
      $this->_helper->viewRenderer->setNoRender();
      $model = new Model_User();
      $users = $model->fetchEntries();
      die(json_encode(array('users' => $users, 'usersCount' =>
count($users))));
    }
    public function getSpecAction() {
      $model = new Model_Specs();
      $requrestedSpec   =   $model->fetchEntry($this->_request-
>getParam('id'));
      die($requrestedSpec['spec']);
    }
    public function saveSpecAction() {
      $this->_helper->viewRenderer->setNoRender();
      Zend_Registry::getInstance()->dbAdapter->query("
          UPDATE
            `specs`
          SET
```

```
                `spec`=\"" . mysql_escape_string(json_decode($this-
>_request->getParam('spec'))) . "\",
                `approved`=1
              WHERE
                `id`=" . (int)$this->_request->getParam('id'));
        }

        public function deleteUserAction() {
          $this->_helper->viewRenderer->setNoRender();
          $model = new Model_User();
          $model->delete((int)$this->_request->getParam('id'));
        }
        public function addUserAction() {
          $this->_helper->viewRenderer->setNoRender();
          $data         =         Zend_Json::decode($this->_request-
>getParam('data'));
          $model = new Model_User();
          $model->save(array(
              'first_name' => $data['first_name'],
              'last_name' => $data['last_name'],
              'email' => $data['email'],
              'status' => !isset($data['is_admin']),
              'company' => $data['company'],
              'occupation' => $data['occupation'],
              'password' => MD5($data['password'])
            ));
        }
    }
```

Фрагмент коду Б.2. Лістинг частини моделі для генерування специфікації вимог

```
    <?php

    class Model_Specs {
        protected $_table;
        public function getTable() {
```

```
            if (null === $this->_table) {
                require_once           APPLICATION_PATH           .
'/Model/DbTable/Specs.php';
                $this->_table = new Model_DbTable_Specs;
            }
            return $this->_table;
        }
        public function save(array $data) {
            $table = $this->getTable();

            $fields = $table->info(Zend_Db_Table_Abstract::COLS);

            foreach ($data as $field => $value) {
                if (!in_array($field, $fields)) {
                    unset($data[$field]);
                }
            }

            return $table->insert($data);
        }
        public function delete($id) {
          $this->getTable()->delete('id = ' . (int)$id);
          //TODO: remove child items : )
        }

        public function fetchEntries($where_condition = '1')
        {
            return    $this->getTable()->fetchAll($where_condition)-
>toArray();
            return $this->getTable();
        }

        public function fetchEntry($id)
        {
            $table = $this->getTable();
            $select = $table->select()->where('id = ?', $id);
```

```
            if($table->fetchRow($select)) {
              return $table->fetchRow($select)->toArray();
            } else {
              throw new Exception("Requested spec doesn't exists");
            }
        }
    }
```

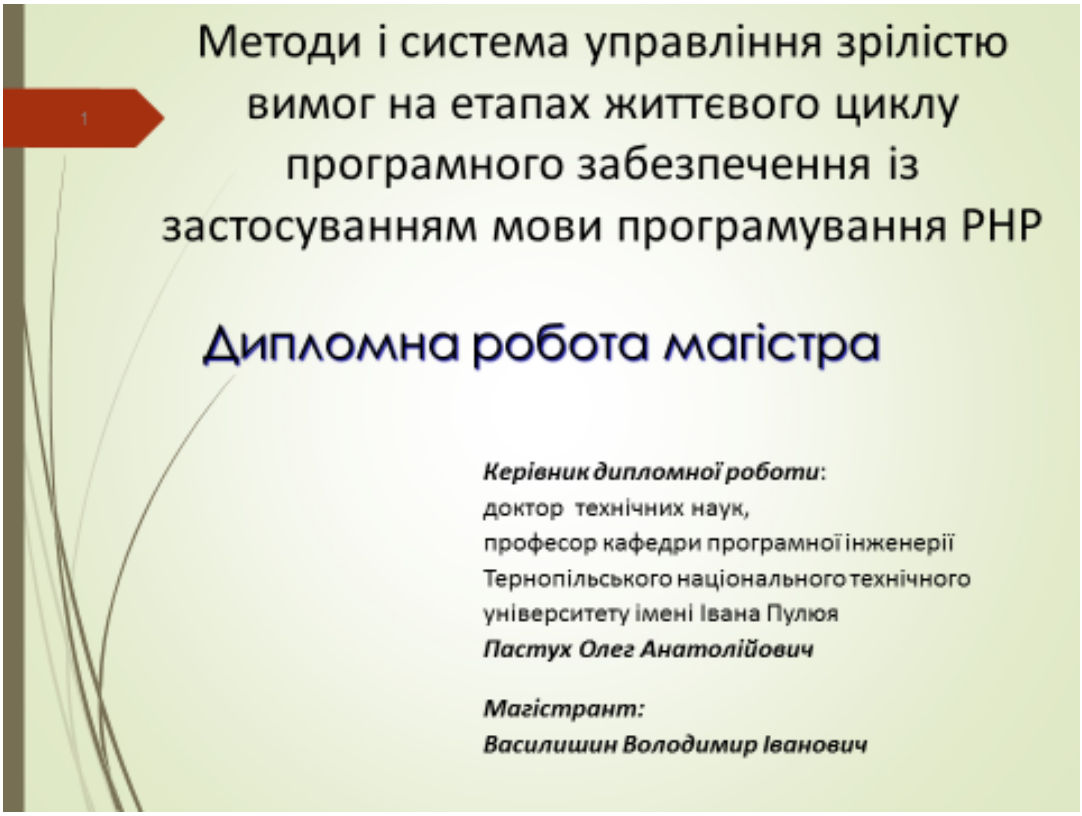
Фрагмент коду Б.3. Лістинг частини View для відображення сторінки генерування специфікації вимог

```
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
      "http://www.w3.org/TR/html4/strict.dtd">

    <html     xmlns="http://www.w3.org/1999/xhtml"     xml:lang="en"
lang="en">
      <head>
        <meta      content="text/html;      charset=UTF-8"      http-
equiv="content-type" />
        <style type="text/css">
          .categories-list {
            list-style: none;
            <?php if($_REQUEST['plain'] == 'true') {  ?>
            padding-left: 0px;
            <?php } ?>
          }
        </style>
      </head>
      <body>
        <?=$this->html; ?>
      </body>
    </html>
```

ДОДАТОК В

Слайди презентації



АКТУАЛЬНІСТЬ ТЕМИ

2

Актуальність теми дослідження пов'язана з необхідністю підвищення ефективності організаційних та основних процесів розробки програмного забезпечення, зокрема на етапах визначення та аналізу вимог до системи.

Важливим завданням галузі інформаційних технологій є забезпечення максимальної зрілості процесів при проектуванні програмного забезпечення. Одним з шляхів підвищення їх якості є впровадження моделей зрілості, які дали б змогу визначити зрілість процесів розробки і самої організації-розробника.

Комплексного підходу щодо забезпечення зрілості процесу розробки вимог до програмного забезпечення, який би давав змогу уніфікувати та кількісно виражати відповідні показники, а також формально описував процедуру забезпечення зрілості процесів програмного забезпечення, на сьогодні не запропоновано. Тому актуальними задачами є побудова моделей зрілості вимог та розробка процедур їх імплементації в реальні проекти розробки програмного забезпечення.

МЕТА, ОБ'ЄКТ І ПРЕДМЕТ ДОСЛІДЖЕННЯ

3

- ***Метою роботи*** є дослідження, обґрунтування та модифікація моделі зрілості вимог при проектуванні програмного забезпечення та перехід на більш якісний рівень розробки вимог до програмного забезпечення.
- ***Об'єктом дослідження є*** процес забезпечення зрілості вимог програмного забезпечення
- ***Предметом дослідження*** є моделі, методи і засоби забезпечення зрілості процесу проектування вимог до програмного забезпечення

ЗАДАЧІ ДОСЛІДЖЕННЯ

4

Для досягнення вказаної мети в роботі поставлено та розв'язано **наступні задачі**:

- аналітичний огляд наукових публікацій і стандартів, що використовуються на етапі виявленні та аналізу вимог до програмного забезпечення;
- обґрунтування та математичне представлення моделі управління зрілістю процесів керування вимогами під час життєвого циклу виконання програмного проекту;
- імплементація процедур підвищення якості рівнів зрілості вимог при управлінні вимогами;
- проектування архітектури підсистеми збору та керування вимогами до програмного забезпечення;
- програмна реалізація прототипу системи збору та управління зрілістю вимог до ПЗ.

МЕТОДИ ДОСЛІДЖЕННЯ

5

Щоб вирішити поставлені задачі було використано наступні методи:

- аналіз та узагальнення – при проведенні аналізу існуючих підходів, моделей, методів і засобів управління зрілістю процесів розробки ПЗ;
- математичного моделювання і теорії множин – при формалізації моделі зрілості вимог і методу підвищення якості її рівнів;
- проектування, програмування і тестування – при реалізації системи управління зрілістю вимог

НАУКОВА НОВИЗНА

6

Наукова новизна одержаних у роботі результатів полягає в наступному:

- уперше запропоновано та модифіковано модель RMM шляхом імплементації рекомендацій стандарту ISO 25010 на вищих рівнях моделі зрілості вимог, що дало змогу забезпечити адекватне їх відображення на наступних стадіях життєвого циклу та оптимізувати ресурси на їх реалізацію, а також реалізувати екосистему CASE-засобів для підвищення ефективності автоматизації процесів життєвого циклу ПЗ;

- набули подальшого розвитку архітектурні та програмні рішення щодо управління зрілістю вимог, які відображають процес розробки програмного забезпечення із застосуванням методології Agile та інтегрованого представлення моделей якості, що дали змогу в повній мірі забезпечити автоматизацію управління зрілістю вимог на основі модифікованої моделі RMM.

Процеси інженерія вимог до ПЗ

7

Перш за все у роботі було проаналізовано процеси, які становлять окремий сегмент в загальному процесі розробки ПЗ. Визначальним при цьому є ядро SWEBOK, що визначає важливі та основні ідеї, концептуальні особливості у сфері інженерії вимог, які продемонстровано нижче.

ПРОЦЕС ПРОЕКТУВАННЯ ВИМОГ ДО ПЗ

8

На початку дослідження було проведено аналіз наукових публікацій та літературних джерел щодо технологій формулювання вимог до програмного забезпечення. На даному слайді наведено результати порівняльного аналізу цих технологій у вигляді таблиці

№ п/п	Метрики	Технології проектування вимог		
		Технологія базована на стандарті IEEE 830	Технологія з використанням шаблонів	Технологія базована на використанні моделі якості
1	Однозначність	3	6	9
2	Завершеність	5	5	9
3	Тестованість	7	7	8
4	Елементарність	3	6	9
5	Швидкість зміни	8	4	6
6	Швидкість видалення	8	4	8
7	Швидкість додавання	7	4	8
Інтегральний показник		0,65	0,57	0,9

ШКАЛА ОЦІНЮВАННЯ ТЕХНОЛОГІЙ ФОРМУЛЮВАННЯ ВИМОГ ДО ПЗ

9

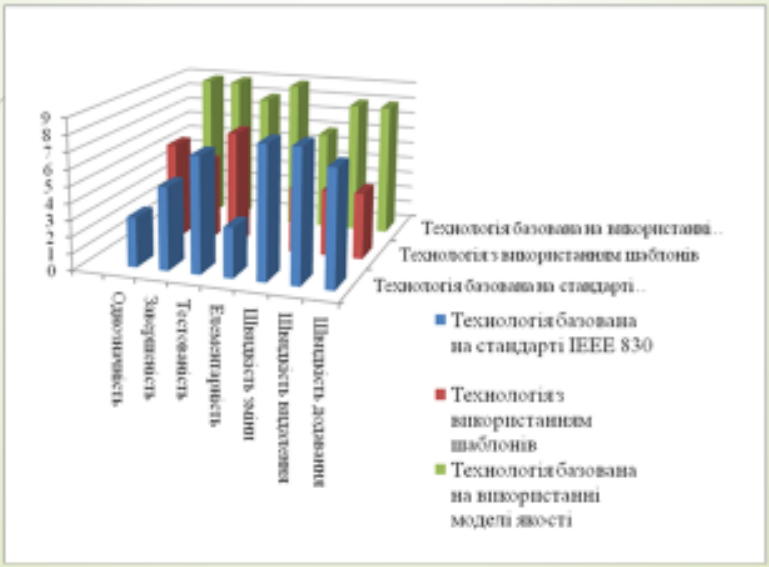
При оцінюванні технологій формулювання вимог до програмного забезпечення використовувалась нечітка шкала, яка наведена у таблиці на даному слайді

Діапазон балів	Нечітке трактування
0-2	Незадовільно
3-5	Задовільно
6,7	Добре
8,9	Відмінно

РЕЗУЛЬТАТИ ОЦІНЮВАННЯ ТЕХНОЛОГІЙ ФОРМУЛЮВАННЯ ВИМОГ ДО ПЗ

10

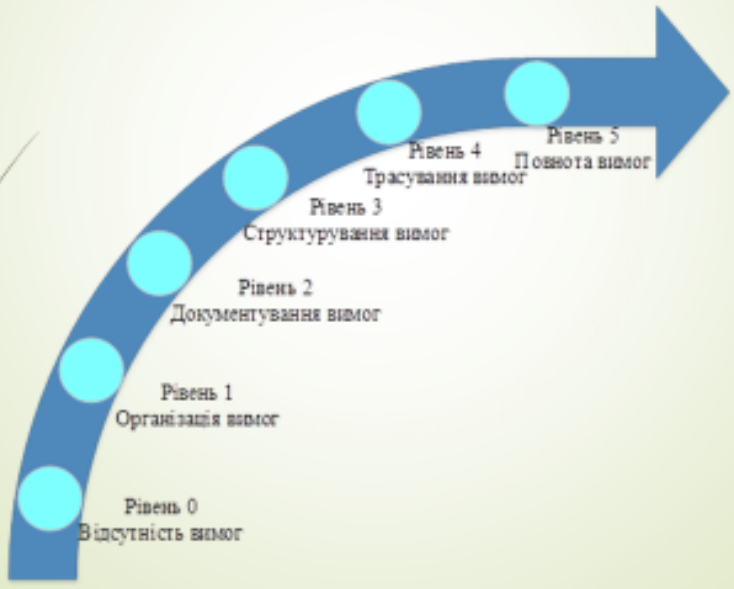
Із наведеної діаграми можна зробити висновок, що найкращою технологією для формулювання вимог є технологія, базована на використанні моделі якості ISO/IEC 25010.



МОДЕЛЬ RMM (REQUIREMENTS MATURITY MODEL)

11

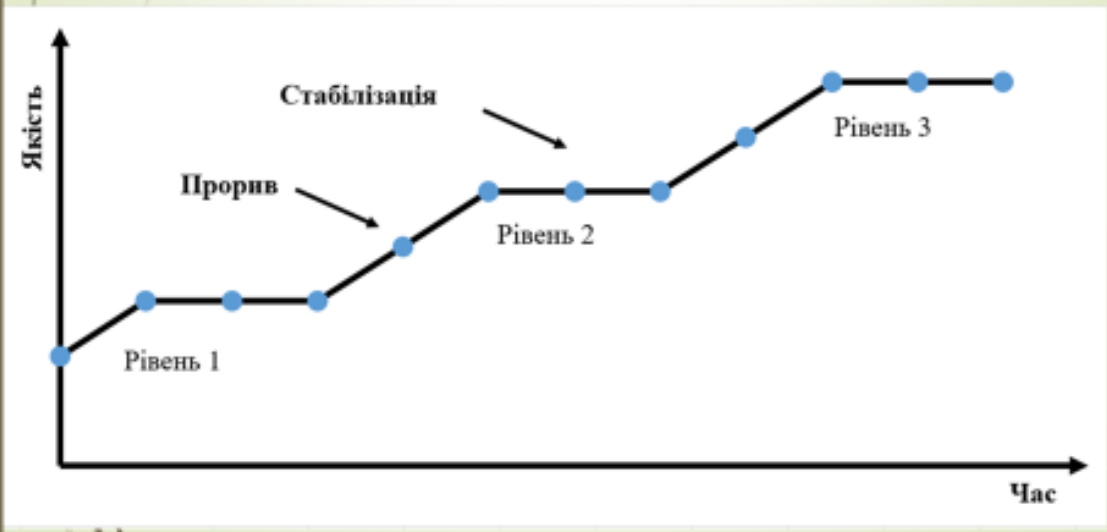
Далі потрібно інтегрувати моделі якості у процеси забезпечення зрілості вимог до програмного забезпечення. Для цього запропоновано використати модель RMM, структуру якої наведено нижче.



ЗРОСТАННЯ РІВНЯ ЯКОСТІ ПРИ ВПРОВАДЖЕННІ МОДЕЛІ RMM

12

Зростання якості процесів і якості продукту спостерігається при переході на вищі рівні моделі RMM і наведено на даному слайді



ХАРАКТЕРИСТИКА РІВНІВ ЗРІЛОСТІ МОДЕЛІ RMM

13

- нульовий рівень зрілості за замовчуванням може бути присвоєний будь-якій команді розробників програмного забезпечення;
- ціль першого рівня зрілості (документування вимог) полягає в отриманні документів, які описують вимоги до програмного забезпечення;
- мета другого рівня зрілості вимог полягає в отримані вимог які зрозумілі замовнику і команді розробника;
- третій рівень зрілості вимог до програмного забезпечення характеризується плануванням процесу керування якістю вимог, класифікації вимог за типами відносно подібності ознак;
- основним завданням четвертого рівня моделі зрілості процесу управління вимогами до програмного забезпечення комп'ютерних систем є визначення і встановлення зв'язків між вимогами;
- основна мета п'ятого рівня моделі зрілості вимог полягає у тому, що вимоги використовуються не тільки для узгодження із замовником, а й для подальшої розробки програмного забезпечення.

ФОРМАЛІЗАЦІЯ МОДЕЛІ ЗРІЛОСТІ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

14

Далі на слайдах зображено математичне представлення моделі зрілості вимог до програмного забезпечення з інтеграцією моделей якості на 3, 4 і 5 рівні.

ФОРМАЛІЗАЦІЯ МОДЕЛІ ЗРІЛОСТІ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Модель зрілості вимог програмного забезпечення

$$RMM = \{L_N\{L_{N-1}\{L_{N-2}\{\dots\{L_0\}\dots\}\}\}\}$$

Нульовий рівень моделі зрілості вимог до програмного забезпечення L_0 **– відсутність вимог**

Перший рівень моделі зрілості вимог програмного забезпечення L_1

$$L_1 = \{Doc_i, ExpMark_i, \{L_0\}\}, i = \overline{1..T} \Rightarrow R_c = \{P_i, C_w\},\ i = \overline{1..I},\quad K = \overline{1..M_i}$$

Doc – множина документів, які описують вимоги до програмного забезпечення, *ExpMark* – експертні оцінки документів, *T* – кількість документів, що описують вимоги на першому рівні моделі зрілості

P_i – потреби користувача;
C_w – обмеження на потреби;
I – кількість потреб замовника;
K – кількість обмежень на потреби.

Другий рівень моделі зрілості вимог до програмного забезпечення, L_2

$$L_2 = \{R_i^c, MClar_{ij}, RClas_{ik}, RVer_{im}, \{L_1\{L_0\}\}\}$$

R_i^c – вимоги замовника до програмного забезпечення зі, $i=\overline{1..B}$, B – кількість вимог на другому рівні моделі зрілості;
$MClar_{ij}$ – методи уточнення вимог (мозковий штурм, анкетування, уточнення, колективна перевірка і т.п.), $j=\overline{1..S}$, S – кількість методів для уточнення вимог;
$RClas_{ik}$ – класи до яких належать вимоги R_i, $k=\overline{1..K}$, K – кількість класів вимог до програмного забезпечення;
$RVer_{im}$ – версії вимоги до програмного забезпечення для контролю внесення у неї змін, $m=\overline{1..M}$, M – кількість версій вимоги.
$RClas = \{Func, NonFunc\}$
Func – клас функціональних вимог, які можна представляти у вигляді шаблонів;
NonFunc – клас нефункціональних вимог до програмного забезпечення на другому рівні моделі зрілості.

ФОРМАЛІЗАЦІЯ МОДЕЛІ ЗРІЛОСТІ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

15

Третій рівень моделі зрілості вимог до програмного забезпечення, L_3

$$L_3 = \{R_i^{***}, Patern_{ij}, RModel_{ik}, RList_{im}, Rcom_h\{L_2\{L_1\{L_0\}\}\}\}$$

R_i^{***} – вимоги замовника у вигляді моделі якості у використанні, $i=\overline{1..G}$, G – кількість вимог якості у використанні;
$Patern_{ij}$ – патерни для опису вимог до програмного забезпечення, $j=\overline{1..Q}$, Q – кількість патернів;
$RModel_{ik}$ – моделі для опису вимог, $k=\overline{1..P}$, P – кількість моделей вимог;
$RList_{im}$ – контрольні листи для перевірки вимог, $m=\overline{1..M}$, M – кількість контрольних листів;
$Rcom_h$ – рекомендації щодо покращення вимог, h – кількість рекомендацій.
$\{P_i, A_{ik}, C_{ik}\},\ i=\overline{1,N},\ K=\overline{1,S_i}$ – множина потреб замовника, атрибутів та обмеження на них;
$\{A_{ik}\},\ K=\overline{1,S_i}$ – множина атрибутів, які відображають ступінь задоволення і-ої потреби;
$R^{***} = \{H_i^*, A_{ik}^*, C_{ik}^*, M_{ik}^*\},\ i \in N_*^k,\ K=\overline{1,S_i}$ – модель якості у використанні;
H_i^* – характеристики моделі якості у використанні;
A_{ik}^* – атрибути моделі якості у використанні;
C_{ik}^* – обмеження на атрибути моделі якості у використанні;
M_{ik}^* – метрики атрибутів моделі якості у використанні;

$$\{P_i, A_{ik}, C_{ik}\},\ i=\overline{1,N},\ K=\overline{1,S_i} \Rightarrow R^{***} = \{H_i^*, A_{ik}^*, C_{ik}^*, M_{ik}^*\},\ i \in N_*^k,\ K=\overline{1,S_i}$$

ФОРМАЛІЗАЦІЯ МОДЕЛІ ЗРІЛОСТІ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

16

Четвертий рівень моделі зрілості вимог програмного забезпечення, L_4

$$L_4 = \{R_i^{ext}, \{L_3\{L_2\{L_1\{L_0\}\}\}\}\}$$

R^{ext} – вимоги до програмного забезпечення у вигляді моделі зовнішньої якості;

$$R^{ext} = \{H_i^x, \Pi_{ik}^x, A_{ik}^x, C_{ik}^x, M_{ik}^x\}, i \in N_x, K = \overline{1, F_i^x}$$

H_i^x – характеристики моделі зовнішньої якості;

Π_{ik}^x – підхарактеристики моделі зовнішньої якості;

A_{ik}^x – атрибути моделі зовнішньої якості;

C_{ik}^x – обмеження на атрибути моделі зовнішньої якості;

M_{ik}^x – метрики атрибутів моделі зовнішньої якості;

П'ятий рівень моделі зрілості вимог програмного забезпечення, L_5

$$L_5 = \{R_i^{in}, \{L_4\{L_3\{L_2\{L_1\{L_0\}\}\}\}\}\}$$

R_i^{in} – вимоги до програмного забезпечення у вигляді моделі зовнішньої якості;

$$R^{in} = \{H_i^y, \Pi_{ik}^y, A_{ik}^y, C_{ik}^y, M_{ik}^y\}, i \in N_x, K = \overline{1, F_i^y}$$

H_i^y – характеристики моделі внутрішньої якості;

Π_{ik}^y – підхарактеристики моделі внутрішньої якості;

A_{ik}^y – атрибути моделі зовнішньої якості;

C_{ik}^y – обмеження на атрибути моделі зовнішньої якості;

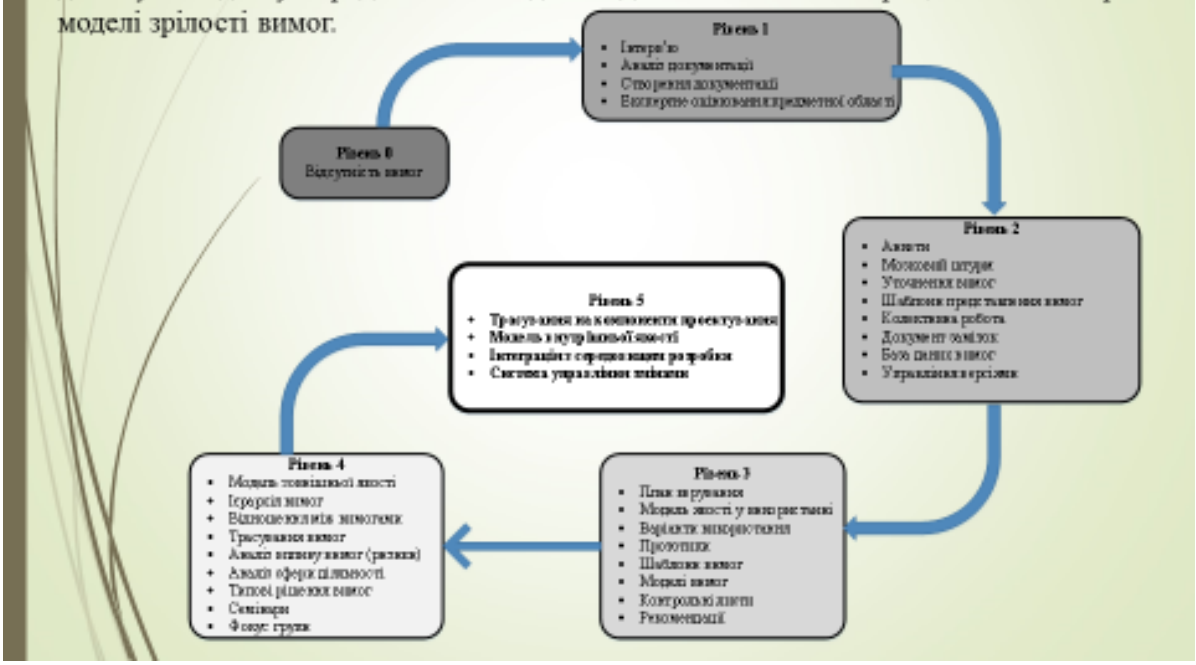
M_{ik}^y – метрики атрибутів моделі зовнішньої якості;

Інтеграція моделей якості дає змогу підвищити якість аналізу та формулювання вимог до програмного забезпечення комп'ютерних систем.

МОДИФІКОВАНА МОДЕЛЬ ЗРІЛОСТІ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

17

У результаті модифікації моделі RMM одержано структуру, яка наведено на даному слайді. Тут представлено модель з детальний описом процесів кожного рівня моделі зрілості вимог.

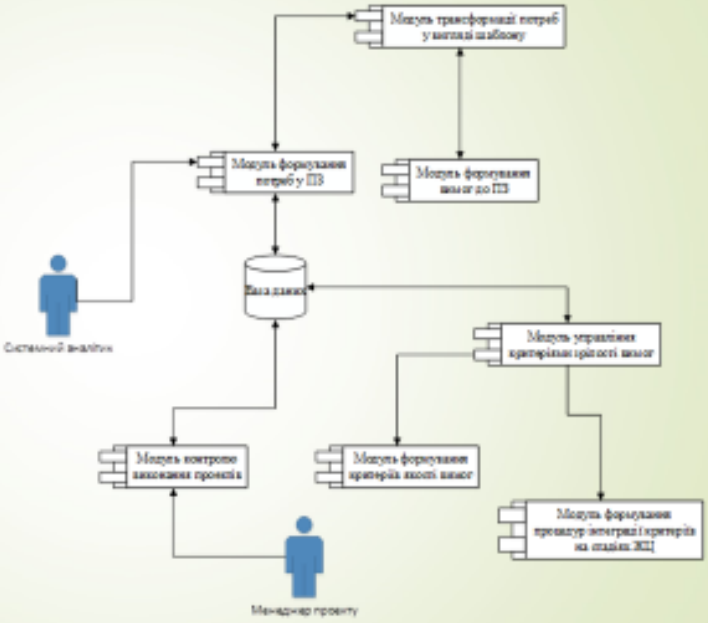


АРХІТЕКТУРА ПРОГРАМНОГО ЗАСОБУ УПРАВЛІННЯ ВИМОГАМИ ДО ПЗ

18

Проаналізувавши існуючи підходи та засоби автоматизації процесом управління вимогами запропоновано архітектуру програмної системи для управління зрілістю на вищих рівнях моделі RMM.

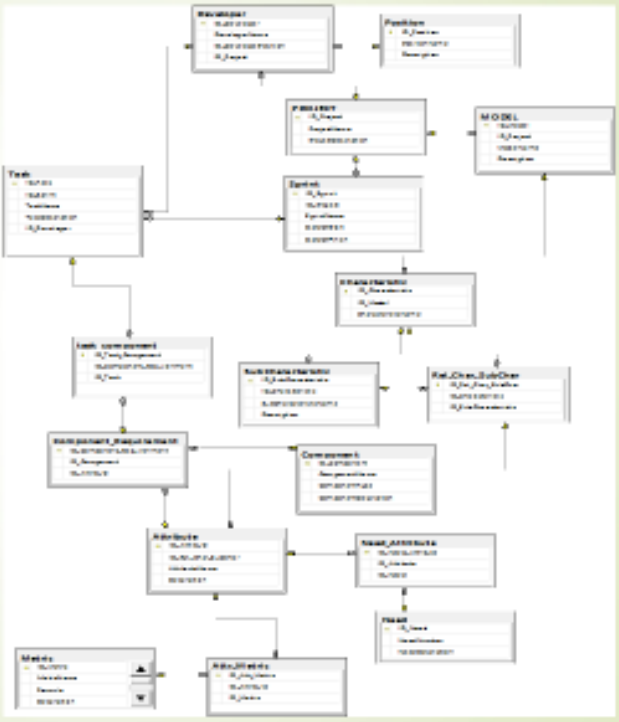
Загальна архітектура системи у вигляді діаграми компонентів програмного засобу на концептуальному рівні, яка представлена на даному слайді



ER-ДІАГРАМА БАЗИ ДАНИХ

19

На даному слайді представлено структурну схему бази даних у вигляді ER-діаграми. У більшості випадків, зв'язки між таблицями мають тип «один-до-багатьох». Типи зв'язків між сутностями «багато-до-багатьох» перетворено шляхом додавання проміжної таблиці та використання типів зв'язків «один-до-багатьох» та «багато-до-одного». Таким чином забезпечено приведення схеми до третьої нормальної форми.

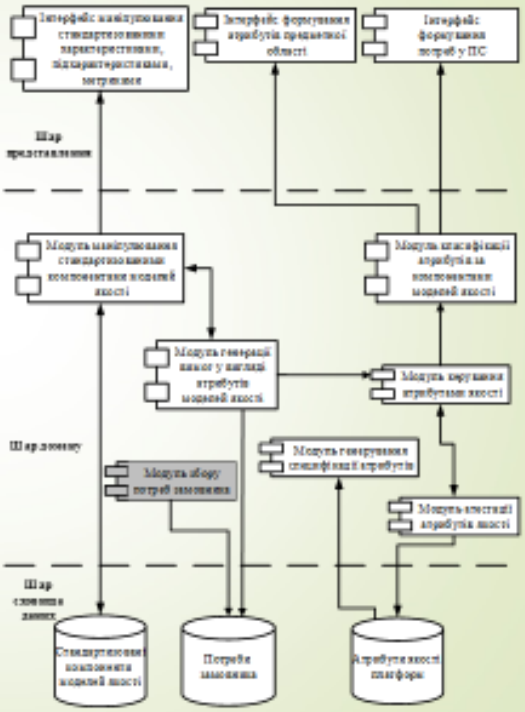


АРХІТЕКТУРА ПРОГРАМНОГО ЗАСОБУ УПРАВЛІННЯ ВИМОГАМИ ДО ПЗ

20

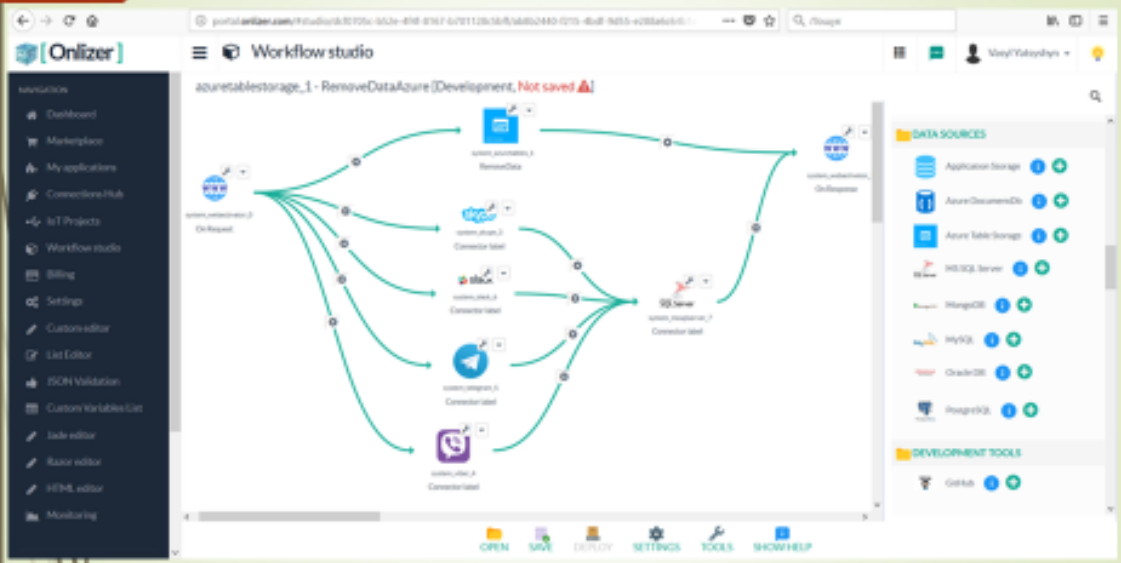
Для оцінювання зрілості вимог програмного забезпечення модифіковано засіб управління вимогами шляхом інтеграції модуля збору потреб замовника з таких джерел як Viber, Skype, Slack і Telegram.

Це дає змогу забезпечити більшу повноту потреб замовника при визначенні вимог до програмного забезпечення комп'ютерних систем.



WORKFLOW ДЛЯ ЗБОРУ ПОТРЕБ ЗАМОВНИКА З ІНТЕРНЕТ КАНАЛІВ ЗВ'ЯЗКУ

21



ВИСНОВКИ

22

Основні наукові та практичні результати роботи полягають у наступному:

1. На основі аналізу основних понять та класифікації вимог до програмного забезпечення визначено особливості процесів інженерії вимог та обґрунтовано актуальність дослідження моделей, методів і засобів управління зрілістю на стадіях життєвого циклу, що в перспективі забезпечить можливість їх оптимізації шляхом підвищення ефективності як самих процесів, так і якості готового програмного продукту.
2. Проведено аналітичний огляд методів та інструментів інженерії вимог у результаті якого визначено критерії якості при формулюванні вимог та підходи до реалізація цього процесу і встановлено, що на сьогодні найбільш ефективними є застосування підходів, які використовуються структуру специфікації рекомендовану стандартом IEEE 830, технологію використання шаблонів на основі структурного та об'єктно-орієнтованого підходів, а також технологію моделей якості.
3. Створено формалізовану схему щодо оцінювання підходів до формулювання вимог ПЗ, яка використовує експертні оцінки і дає змогу кількісно представити ефективність їхнього застосування в задачах забезпечення та управління зрілістю вимог на стадіях життєвого циклу. Встановлено, що кращим варіантом є використання моделей якості для представлення вимог до ПЗ.

ВИСНОВКИ

23

4. Обґрунтовано модель зрілості RMM (Requirements maturity model) для управління зрілістю вимог до програмного забезпечення, що дало змогу підвищити якість процесу виявлення, аналізу та керування вимогами і забезпечити швидкий перехід команди-розробників з нульового на вищі рівні моделі зрілості.
5. Запропоновано модифікацію моделі RMM шляхом імплементації рекомендацій стандарту ISO 25010 на вищих рівнях моделі зрілості вимог, що дало змогу забезпечити адекватне їх відображення на наступних стадіях життєвого циклу та оптимізувати ресурси на їх реалізацію.
6. Формалізовано модифіковану модель зрілості вимог за допомогою апарату теорії множин та забезпечено вкладеність рівнів з безшовним переходом між ними, що дало змогу в подальшому запропонувати екосистему CASE-засобів та підвищити ефективність автоматизації процесів життєвого циклу ПЗ.
7. Проведено аналіз сучасних інструментів підтримки процесів життєвого циклу ПЗ у результаті якого виявлено їхні недоліки та здатність до взаємодії з іншими засобами автоматизації, що дало можливість виявити потенційні шляхи впровадження модифікованої моделі зрілості вимог.

ВИСНОВКИ

24

5. Уперше обгрунтовано та модифіковано модель зрілості вимог RMM шляхом інтеграції на третій, четвертий і п'ятий рівень моделей якості стандарту ISO/IEC 9126, що дало змогу підвищити якість виконання процесу розробки вимог та знизити складність переходу організацій-розробників з одного рівня зрілості на інший.
6. Уперше, на основі апарату теорії множин, формалізовано модель зрілості вимог до програмного забезпечення при проектуванні комп'ютерних систем, що дало змогу автоматизувати процеси трасування та управління вимогами і як наслідок підвищити ефективність виконання проектів з побудови комп'ютерних систем.
7. На основі аналізу модифікованої моделі зрілості вимог програмного забезпечення при проектуванні комп'ютерних систем визначено процеси, які необхідно автоматизувати для підвищення якості розробки вимог та проведено аналіз існуючих CASE засобів розробки вимог, що дало змогу виявити потенційну можливість їх адаптації до підтримки запропонованої моделі зрілості вимог.

ВИСНОВКИ

25

8. Спроектовано архітектуру програмної системи управління зрілістю вимог, що відображає процес розробки програмного забезпечення із застосуванням методології Agile та інтегрованого представлення моделей якості, що дало змогу в повній мірі забезпечити автоматизацію управління зрілістю вимог на основі модифікованої моделі RMM.
9. Розширено можливості щодо збору потреб замовників з сучасних каналів комунікації шляхом використання інтеграційної платформи Onlizer, що дало змогу підвищити рівень повноти вимог та автоматизувати початкові рівні управління зрілості вимог.


26
ДЯКУЮ ЗА УВАГУ!!!