Дотримання RMS інструментами вимог якості SRS та дизайн комплементарної системи

https://doi.org/10.31713/MCIT.2024.041

Роман Крук
Національний університет водного господарства та природокористування, НУВГП
Рівне, Україна
r.a.kruk@nuwm.edu.ua

Наталія Жуковська
Національний університет водного господарства та природокористування, НУВГП
Рівне, Україна
n.a.zhukovska@nuwm.edu.ua

Анотація – **У цій роботі досліджуються виклики інженерії вимог до програмного забезпечення в Agile-проєктах. Розглянуто проблеми, пов'язані з неповними, неконсистентними та неструктурованими вимогами, і підкреслено, як ітеративна природа Agile ускладнює управління вимогами. Дослідження порівнює широко використовувані інструменти, такі як Jira та Confluence, в управлінні вимогами до програмного забезпечення, виявляючи прогалини в їх здатності відповідати ключовим атрибутам якості. У роботі також представлено концептуальну модель для покращення управління вимогами завдяки динамічній системі моделювання SRS.**

Ключові слова – **Agile; інженерія вимог; вимоги до ПЗ; атрибути якості SRS; специфікація до ПЗ; системний аналіз; міждисциплінарний підхід.**

I. Вступ

Гнучка методологія розробки ПЗ, хоч і найефективніша у реаліях сучасного мінливого ринку, створює також унікальні виклики у сфері інженерії вимог до ПЗ та управління ними[1]. Порівняльне дослідження українського IT-сектора викрило наступні основні проблеми управління вимогами до ПЗ в Agile-проєктах:

1.Пропущені або неповні вимоги до ПЗ;

2.Неконсистентність вимог до ПЗ;

3.Брак простежуваності вимог до ПЗ та можливостей виявлення залежностей між ними;

4.Низький рівень структурованості вимог до ПЗ та вичерпності специфікації до продукту;

Усі з вище перелічених викликів спричинені особливостями методології Agile, а саме - ітеративного підходу до опрацювання вимог до ПЗ. У класичних методологіях розробки, зокрема водоспадній моделі, специфікація вимог для ПЗ формується вичерпною ще до початку будь-яких інженерних робіт [2], таким чином уникаючи вище перелічених викликів, а втім така модель є маложиттєздатною на сучасному ринку [3][4], за винятком проєктів, що вимагають виключної завершеності (наприклад, ПЗ для будівництва, транспорту, медицини тощо) [5].

Дане дослідження ґрунтується на спробі авторів поглянути на інженерію вимог і, відповідно, проблеми, які наявні в цій сфері, як на процес повністю самостійний та відокремлений від загального "управління вимогами", "управління IT-проєктом" чи "дизайну системи". Відповідно, з цієї точки зору етап інженерії вимог потребує окремої повноцінної прикладної системи для підтримання високого рівня якості вимог до ПЗ протягом усього життєвого циклу IT-проєкту, схожого до того, який існує на рівні загального "управління проєктом" (веб-платформи Atlassian Jira та Confluence), а також для "управління програмним кодом" на прикладі CI/CD та технології Git.

Таким чином, мета даного дослідження полягала в аналізі актуальності та важливості атрибутів якості вимог до ПЗ, а також ступеня відповідності поточних інструментів управління вимогами, поширених у проєктах гнучкої методології розробки з рекомендованими атрибутами якості вимог для виявлення слабких місць. Для цього був проведений:

1.Аналіз професійної та наукової літератури для агрегації знань щодо рекомендованих атрибутів якості вимог до ПЗ;

2.Аналіз Atlassian Jira та Confluence як поширених в Agile-проєктах інструментів управління вимогами та огляд їх основних функцій та принципу роботи;

3.Аналіз відповідності атрибутів якості вимог до ПЗ з функціями інструментів управління вимогами до ПЗ;

Формування висновків та подальших кроків дослідження.

II. Літературний огляд

Для аналізу літератури використовувався метод систематичного огляду літератури (СОЛ), який дозволяє досліджувати, оцінювати та підсумовувати попередні дослідження. Цей метод включає три етапи: планування, аналіз та звіт. На етапі планування було розроблено дослідницькі питання, що стосувалися якості вимог до ПЗ, зокрема критеріїв якості для специфікації вимог у контексті

Agile-проєктів. Процес пошуку літератури відбувався через академічні бази даних, такі як Google Scholar, IEEE Explore, ACM Digital Library та інші. Було визначено критерії включення і виключення публікацій, зокрема відокремлено роботи, що стосуються якості вимог, але не якості самого ПЗ. Публікації, що не відповідали цим критеріям, були відсіяні, і було відібрано 261 первинне дослідження, з яких 16 відповідали темі дослідження.

Однією з основних проблем, виявлених у літературі, є складнощі з визначенням повних і узгоджених вимог до ПЗ. Часто клієнти не можуть точно сформулювати всі вимоги на початкових етапах проєкту, що призводить до виникнення проблем із відстеженням вимог і повторною розробкою. Для вирішення цих проблем необхідні додаткові дослідження. Аналіз також охоплює методи перевірки якості вимог, більшість з яких належать до знання-орієнтованих підходів. Однак інструменти для перевірки вимог часто обмежені, і багато з них залежать від специфіки домену. Виникають труднощі з вираженням вимог і отриманням зворотного зв'язку від клієнтів, що ускладнює практичне застосування цих методів. У результаті проведеного огляду було виявлено 20 атрибутів якості вимог до ПЗ та 11 атрибутів специфікації. Атрибути були систематизовані та ранжовані на основі поширеності в літературі. Зібраний матеріал дозволив виявити прогалини у дослідженнях та сфокусувати подальші дослідження на вирішенні цих проблем.

III. Порівняльний аналіз інструментів управління вимогами в Agile

Під час проведеного аналізу публікацій та фахових посібників було визначено основні категорії атрибутів якості, встановлено їхні визначення та поширеність. На основі проведеного аналізу було визначено одностайність або сумнівність кожного із наведених атрибутів якості.

Було визначено, що основними затребуваними атрибутами якості специфікації вимог до ПЗ є структурованість, завершеність. Узгодженість, простежуваність, змінюваність, верифікабельність та ненадмірність (Рис. 1).

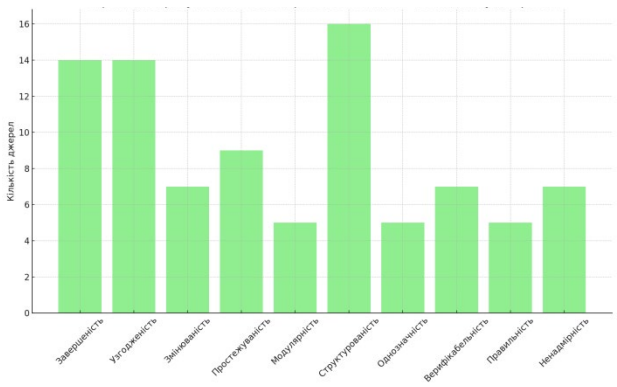

Рисунок 1. Атрибути якості специфікації вимог до ПЗ згідно зі згадками в проаналізованій літературі

Наступний етапом дослідження був порівняльний аналіз найпоширеніших Agile-інструментів для управління вимогами. Agile-інструмент - це будь-яка інформаційна система, що поширено використовується в IT Agile-проєкті для роботи з певною частиною проєктної інформації, що містить вимоги до ПЗ.

Більшість інструментів управління вимогами належать до одного з двох зазначених типів: Issue Tracking Software (далі - ITS) або ж Wiki-Software (далі - WS). Таким чином, методом зворотньої інженерії можна виснувати, що ці дві категорії інформаційних систем, що так поширено використовуються в Agile-проєктах, задовольняють дві базові потреби:

1. ITS дозволяють керувати мінливим та ітеративним процесом розробки ПЗ, зосереджуючись перш за все не на вимогах, а на завданнях, що повинні бути виконані. Такі завдання не обов’язково корелюються з вимогами нового чи вдосконаленого функціоналу;
2. WS, на противагу, виконують роль сховища проєктної специфікації в цілому, та специфікації вимог до ПЗ зокрема.

Втім саме ITS є основними у використанні в Agile-проєктах, тимчасом як WS виконують лише допоміжну роль. З іншого боку, саме WS за належного користування ними, могли б забезпечити вирішення викликів, перелічених у Вступі даної роботи. Ручна синхронізація вимог в ITS та специфікації в WS є часозатратною і такою, що не дає переваг у близькій перспективі. Стратегічною ж перспективою часто нехтують. Разом з тим, наразі відсутнє рішення, що забезпечувало б синхронізацію між вимогами до ПЗ з ITS та WS системи, в силу складнощів їх неструктурованої природи. Для подальших пошуків, було проаналізовано, як такі два типи системи працюють зі своїми вимогами.

ITS мають кілька спільних ключових функцій, які спрощують процес виявлення, реєстрації та керування завданнями в рамках проекту. ITS дозволяють користувачам систематично повідомляти про проблеми чи помилки, надаючи атрибути для детального опису, рівнів серйозності, пріоритету та вкладень, класифікувати проблеми, сортувати та фільтрувати за різними критеріями [6; 7]. ITS підтримують конфігуративні робочі процеси, які зображають життєвий цикл проблеми, від створення до вирішення. ITS полегшують спілкування між членами команди розробки за допомогою вбудованих систем комунікації. ITS інтегруються з іншими інструментами розробки та управління проектами, такими як системи контролю версій Git, конвеєри безперервної інтеграції/безперервного розгортання (CI/CD) і програмним забезпечення для управління проектами. ITS забезпечують можливості звітності та аналітики, такими як час вирішення завдання, розподіл робочого навантаження та тенденції виникнення проблем. ITS включають контроль доступу на основі ролей, щоб гарантувати доступ до даних лише авторизованого персоналу. ITS ефективні для відстеження завдання і керування їх

прогресом, але часто не справляються з ефективним керуванням вимогами до ПЗ.

Щодо недоліків ITS:

1. Відсутність функцій керування вимогами: ITS не вистачає необхідних полів і структур для обробки складних вимог, таких як залежності, критерії прийняття та детальні специфікації, а також належного формування цілісної, вичерпної та несуперечливої специфікації вимог до ПЗ, необхідної для ефективного управління IT-продуктом і його подальшого розвитку. Як вже було проаналізовано в попередніх роботах, особливо цей недолік починає проявлятися в довготривалих “спадкових” проєктах, де інженери та аналітики вимог можуть змінюватись з часом;
2. Невідповідна відстежуваність: ефективне управління вимогами вимагає надійної відстежуваності між різними компонентами системи на рівні вимог до даних та процесів, що далеко не завжди може бути ефективно забезпечено ITS системою;
3. Погана підтримка ієрархічних вимог: вимоги до ПЗ потрібно організовувати ієрархічно, при цьому вимоги високого рівня розбиваються на підвимоги. ITC добре підтримує ієрархічну структуру, але при цьому орієнтована не на вимоги до ПЗ, а на структурні одиниці роботи, які не обов’язково відповідають вимогам до ПЗ. Так, згідно з відкритою структурою БД застосунку Jira [8] чітко простежується задачо-орієнтованість, а не вимого-орієнтованість. Часткова узагальнена діаграма “сутність-зв’язок” наведена на Рис. 2 нижче.

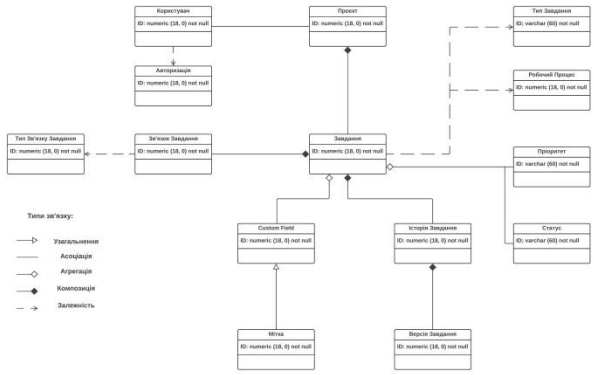

Рисунок 2.Часткова узагальнена діаграма “сутність-зв’язок” домену Confluence Jira запроєктована методом зворотньої інженерії

В проєктах гнучкої методології розробки використовуються також WS. Системи Wiki — це платформи для спільної роботи, які дозволяють користувачам спільно створювати, змінювати та організовувати вміст. Одним із яскравих прикладів є Atlassian Confluence, широко використовувана корпоративна вікіпедія, яка підтримує різноманітну професійну та академічну діяльність[9].

Atlassian Confluence пропонує ряд функцій, які роблять його надійним інструментом для управління знаннями та співпраці:

1.Створення сторінок і блогів: користувачі можуть створювати та редагувати сторінки та блоги за допомогою форматованого текстового редактора, що дозволяє легко документувати та ділитися знаннями;

2.Макроси та розширення: Confluence підтримує макроси та розширення, які покращують функціональність, наприклад динамічне вбудовування вмісту, діаграм і списків завдань;

3.Інтеграція з іншими інструментами як Atlassian (наприклад, Jira), так і програмами сторонніх розробників;

4.Контроль версій;

5.Система авторизації;

IV. Прототипування динамічної системи моделювання SRS

Попередні роботи приділяли недостатню увагу моделі даних, яка б найкраще підходила для забезпечення потреб атрибутів якості вимог до ПЗ. Метою даної частини дослідження була проєктування моделі даних та функціоналу динамічної системи моделювання SRS. Для цього було прийнято рішення використати методи моделювання даних та прототипування, описані у БАБОК [10, с 256][10, с 323]. Метод мови моделювання онтологій UML використовує уніфіковану мову моделювання (UML) для представлення онтологій, які є формально заданими моделями, що визначають концепції та зв’язки в межах домену. Авторами роботи було прийнято рішення у аналізі продовжувати використання дедуктивного підходу до дослідження. Було узгоджено, що подальше прототипування має базуватися на наступних припущеннях:

1. У даній моделі функціональна вимога прийматиметься за цілісну структурну одиницю, що не підлягає подальшій декомпозиції;
2. Дана модель буде побудована згідно з дата-орієнтованого підходу, оскільки кінцева мета напрямку досліджень - оптимізація інформації та знаннях, які зберігаються у специфікаціях вимог до ПЗ в Agile-проєктах, аби завдяки цьому фактору підвищити ефективність таких проєктів у довготерміновій перспективі.

Принципи такого підходу включають включають:

1. Зведення до мінімуму переміщення даних;
2. Максимальне перевикористання: дизайн, орієнтований на дані, має на меті максимізувати повторне використання даних, оскільки це може призвести до більш ефективного використання пам’яті та інших ресурсів.

Перевагою такого підходу є орієнтація на покращення продуктивності та гнучкості як самої системи, так і процесу проєкту, де вона застосовна. Було обрано ітеративний підхід до моделювання. З поточної моделювання було повністю виключено великі мовні моделі та інші технології, що

орієнтуються на обробку натуральних мов, оскільки фокус на обробку даних поза обсягом робіт даного моделювання.

Проєктування включало міждисциплінарний підхід, а саме поєднання системного аналізу та комп'ютерних наук. Було розроблено модель використання системи (див. Рис. 3), а також високорівнева модель розгортання системи (Рис. 4)

Рисунок 3. Діаграма варіантів використання динамічної системи моделювання SRS

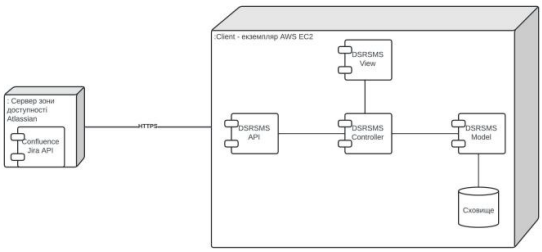

Рисунок 4. Високорівнева діаграма розгортання динамічної системи моделювання SRS

V. Висновки

У цій частині роботи підсумовуються результати проведеного дослідження та аналізу, а також окреслюються подальші питання та кроки дослідження. Метою першого етапу даної роботи було провести огляд літератури та публікацій на предмет атрибутів якості вимог до ПЗ. На другому етапі дослідження був проведений огляд двох основних типів систем, які використовуються на управління вимогами: Issue Tracking Software та Wiki-software. Аналіз показав, що функціоналу обох типів застосунків притаманна деревовидна структура даних у контексті вертикальної ієрархії вимог, і звичайного напрямленого розрідженого графу з циклами - для горизонтальних зв'язків. Дана властивість відповідає вимогам до структурованості та модулярності SRS. Втім, всі вище перелічені якості стосуються сутностей-носіїв вимог. Вони не у повній мірі поширюються на самі вимоги до ПЗ. Такі ж якості вимог, як атомарність, змінюваність не мають функціоналу на їх задовільну підтримку, що впливає на кінцеву якість специфікації.

Перевагою даного дослідження є порівняльний аналіз функціоналу поширених систем управління вимогами на наявність інструментів для задоволення атрибутів якості вимог до ПЗ та проєктування моделі системи, яка б була здатна виправити наявні недоліки. Недоліком даного дослідження є, безумовно, вузька вибірка систем для аналізу, що обмежується поширеними в Agile-проєктах України Atlassian Jira та Confluence.

Таким чином, подальшими кроками в дослідженні буде розробка системи управління вимогами до ПЗ, котра відповідатиме всім атрибутам якості вимог до ПЗ, описаним у даній роботі. На основі проведеного аналізу автори висувають припущення, що за основу даної системи є доцільним використати вертикальну деревовидну структуру даних, притаманну системам Jira та Confluence, а також ввести додатковий параметр до графу, для забезпечення аудиту та версіонування кожного його вузла. Робочою моделлю такої системи стане тривимірний граф, вузлами якого будуть безпосередньо вимоги до ПЗ, а не їх сутності-носії. Також доцільним брати за основу простий орієнтований граф для позначення зв'язків між вимогами, аби забезпечити відстежуваність. Для модулярності планується використати мітки категоризації. Окремим питанням до подальшого дослідження є перетворення вимог до ПЗ з природної мови у структуровані або напівструктуровані дані.

Література

[1] Kruk, R., & Zhukovska, N. (2023). SURVEY STUDY OF REQUIREMENTS ENGINEERING ISSUES AND CHALLENGES IN AGILE PROJECTS. Вісник НУВГП. Технічні науки, 103(3), 195–212.

[2] Adenowo, A., & Adenowo, B. A. (2020). Software Engineering Methodologies: A Review of the Waterfall Model and Object-Oriented Approach. International Journal of Scientific and Engineering Research, 7(4), 427–434.

[3] Pargaonkar, S. (2023). A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering. International Journal of Scientific and Research Publications, 13(8), 120–124.

[4] Thesing, T., Feldmann, C., & Burchardt, M. (2021). Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project. CENTERIS - International Conference on ENTERprise Information Systems, 746–756.

[5] El-Sokkary, N., El-Masry, W. H., & Darwish, N. R. (2021). A Proposed Hybrid Approach for Developing Healthcare Information Systems. International Journal of Computer Applications, 183(28), 17–23. https://doi.org/10.5120/ijca2021921664

[6] Jira Cloud resources | Jira Cloud | Atlassian Support. Atlassian Support. URL: https://support.atlassian.com/jira-software-cloud/resources/ (date of access: 13.08.2024).

[7] Trello Guides: Help Getting Started With Trello | Trello. Manage Your Team's Projects From Anywhere | Trello. URL: https://trello.com/guide (date of access: 13.08.2024).

[8] Database schema. Atlassian Developer. URL: https://developer.atlassian.com/server/jira/platform/database-schema/ (date of access: 13.08.2024).

[9] Confluence Cloud resources | Confluence Cloud | Atlassian Support. Atlassian Support. URL: https://support.atlassian.com/confluence-cloud/resources/ (date of access: 13.08.2024).

[10] 18.IIBA. (2015). A Guide to the Business Analysis Body of Knowledge (BABOK Guide) Version 3.0. International Institute of Business Analysis.