Науковий вісник НЛТУ України
Scientific Bulletin of UNFU
https://nv.nltu.edu.ua
https://doi.org/10.15421/40280727
Article received 22.06.2018 p.
Article accepted 26.09.2018 p.
УДК 004.[891.3+3]:681.322[075.8]
ISSN 1994-7836 (print)
ISSN 2519-2477 (online)
@ Correspondence author
Yu. I. Hrytsiuk
yurii.i.hrytsiuk@lpnu.ua

Ю. І. Грицюк, О. А. Нємова
Національний університет "Львівська політехніка", м. Львів, Україна

ОСОБЛИВОСТІ ФОРМУЛЮВАННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Запропоновано мовні особливості процедури формулювання вимог до програмного забезпечення (ПЗ), які є загальними для будь-якого процесу його розроблення, що дає змогу аналітику на підставі їхніх можливостей з'ясувати особливості розроблення структури відповідного документа та здійснити відбір ключових вимог з множини допустимих. Проаналізовано основні можливості вимог до ПЗ та їхні атрибути, жорстко прив'язані до цих вимог, що допомогло зрозуміти суть пропонованих принципів формулювання вимог до ПЗ. З'ясовано особливості розроблення структури документа з вимогами до ПЗ та деякі моменти відбору ключових вимог, що дає змогу його виконавцям доступно викладати в ньому потрібний матеріал з завершеністю думки, здійснювати логічну послідовність подання інформації, забезпечувати її цілісність й доступність, ясність для подальшого використання та сприйняття. Встановлено взаємопов'язаність та важливість вимог до ПЗ шляхом їх класифікації, фільтрування й сортування для того, щоб згодом отримати відносно невеликі набори вимог, кожен з яких стосується тільки однієї теми для подальшого їх аналізу. Виявлено мовні особливості процедури формулювання вимог до ПЗ та уточнено деякі моменти підготовки відповідних шаблонів і їхньої деталізації, які сукупно ведуть до розроблення якісного ПЗ шляхом тісної співпраці бізнес-аналітика з його замовником, внаслідок якої вони встановлюють і записують відповідні домовленості. Проаналізовано критерії, які потрібно використовувати для визначення якості формулювання вимог до ПЗ, що дає змогу аналітику дотримуватися деяких простих правил при їх написанні.

Ключові слова: програмний проект; вимоги до програмного забезпечення; визначення вимог; розроблення вимог; формулювання вимог; рецензування вимог; специфікація вимог; критерії якості вимог; структура документа.

Вступ. Вимоги до програмного забезпечення (англ. *Software Requirements*) – це набір потреб потенційних користувачів щодо властивостей, якості та функцій програмного продукту, який потрібно розробити або модифікувати (Wiegers, 2003). *Визначення вимог* (англ. *Requirements Definition*) – це процедура витягування інформації з різних джерел (договорів, матеріалів аналітиків, шляхом декомпозиції завдань та функцій системи й ін.), проведення технічних заходів (співбесід, інтерв'ю, виробничих нарад і ін.) для формування окремих наборів вимог до майбутнього програмного продукту (Hrytsiuk, 2018).

Однак, процедура *формулювання вимог* (англ. *Requirements Formulation*) докорінно відрізняється як від процедури визначення вимог до ПЗ, так і від процедури написання програмних кодів. Ця процедура абсолютно не схожа на написання романів, творів чи сценаріїв. Вона також не схожа і на процедуру підготовки звичайної технічної документації, наприклад, такої, як настанова з експлуатації ПЗ або настанова користувача (Wiegers & Betti, 2014). Водночас, процедура формулювання вимог до ПЗ є невід'ємною складовою процесу розроблення вимог до ПЗ. Адже, *розроблення вимог* – це цілий технологічний процес, який має свої етапи та певні особливості його реалізації, результатом якого є підготовлена специфікація вимог до ПЗ, в якій зазначено, що за функціональні та не функціональні можливості має мати майбутній програмний продукт (Dick, Hull & Jackson, 2017).

Під час формулювання вимог до ПЗ аналітику потрібно акуратно збалансувати такі два надзвичайно важливі принципи (Hrytsiuk, 2018):

- *структурованість документа,* тобто документ з вимогами до ПЗ має бути зручним для читання учасниками проекту та зрозумілим для зацікавлених сторін – як замовника ПЗ, так і його розробників;
- *якість формулювання окремої вимоги*, тобто сформульована вимога до ПЗ має бути не тільки зручною для подальшої роботи з нею, але й забезпечувати якість процесу розроблення програмного проекту.

Під першим принципом потрібно розуміти, що документ з вимогами до ПЗ має бути структурованим так, щоб замовнику ПЗ чи його розробникам було легко розуміти формулювання кожної індивідуальної вимоги як всередині розділу, так і в усьому документі. Стосовно другого принципу, то тут йдеться, насамперед, про якість формулювання кожної окремої вимоги до ПЗ, тобто, чи усім учасникам проекту зрозумілою мовою вона написана чи внаслідок її виконання отримане ПЗ буде мати належну якість. Тут також потрібно звернути увагу на те, наскільки такий опис чітко і точно відображає суть реальної потреби замовника ПЗ, його безпосе-

Інформація про авторів:

Грицюк Юрій Іванович, д-р техн. наук, професор, кафедра програмного забезпечення. **Email:** yurii.i.hrytsiuk@lpnu.ua; https://orcid.org/0000-0001-8183-3466; ResearcherID: V-3995-2017

Нємова Олена Андріївна, студентка, кафедра програмного забезпечення. **Email:** olena.nemova@gmail.com

Цитування за ДСТУ: Грицюк Ю. І., Нємова О. А. Особливості формулювання вимог до програмного забезпечення. Науковий вісник НЛТУ України. 2018, т. 28, № 7. С. 135–148.

Citation APA: Hrytsiuk, Yu. I., & Nemova, E. A. (2018). Peculiarities of Formulation of Requirements to the Software. *Scientific Bulletin of UNFU, 28*(7), 135–148. https://doi.org/10.15421/40280727

редніх користувачів чи конкретного завдання для його розробників, наскільки вимогу можна подавати у вигляді деякого об'єкта, з яким зручно встановлювати зв'язки від попередніх і до наступних вимог до ПЗ.

Досвідчені фахівці, які працюють над формулюванням вимог до ПЗ, добре розуміють, що звичайного текстового редактора не завжди достатньо для організації процесу управління вимогами, створення їхньої структури, класифікації вимог, а також визначення як їх властивостей, так і встановлення зв'язків між ними. Для розуміння обмежених можливостей текстового редактора можна навести приклад, в якому нумерують вимоги згідно з номерами розділів відповідного документа. Спроба вставити всередину документа нову вимогу призведе до того, що всі наступні вимоги автоматично мають змінити свої номери.

З аналогічними незручностями стикаються й ті аналітики, які для підготовки вимог до ПЗ намагаються використовувати бази даних. Вони згодом доходять висновку, що наявність численних таблиць, заповнених індивідуальними вимогами, також практично не дає змоги управляти ними як єдиною та цілісною структурою. Тобто незважаючи на те, що за допомогою бази даних легко нумерувати, класифікувати й сортувати вимоги до ПЗ, життєво важливий сенс всього документа можна втратити, якщо деяку конкретну вимогу вилучити із загальної структури документа (Hrytsiuk & Leshkevych, 2017).

Рецензування вимог (англ. *Requirements Review*) – перевірка вимог (а також відповідних документів, концепцій, технічного завдання і т.д.) більш досвідченими фахівцями (експертами) на предмет виявлення неповноти вимог у їх наборах, неточностей їх формулювання або суперечностей між ними, неправильних атрибутів чи обмежень і т.д. Мета цієї процедури – поліпшити структуру вимог до ПЗ та їх якісний склад, покращити стиль написання вимог аналітиками і, що найважливіше, зміст їхніх формулювань, зрозумілим як для замовника ПЗ, так і його розробників. Тут важливо не плутати зазначене з процедурою узгодження вимог із зацікавленими сторонами (Hrytsiuk, 2018).

Наведені вище принципи формулювання вимог до ПЗ – *структурованість документа* та *якість формулювання окремої вимоги* – мають постійно перебувати під пильною увагою як аналітиків, так і розробників ПЗ (кодувальників і тестувальників), а також керівника проекту. При цьому багато науковців і практиків вважають, що процедури формулювання вимог до ПЗ та їх рецензування кваліфікованими експертами потрібно проводити паралельно. Це має проходити хоча б з тієї простої причини, що критерії, які використовують для визначення якості формулювання вимог до ПЗ, ті самі, що і для їх прискіпливого рецензування. Саме тому деякі особливості формулювання вимог до ПЗ та їх рецензування розглядатимемо нижче разом у цьому дослідженні.

Аналіз останніх досліджень та публікацій. Проблемі визначення вимог до ПЗ присвячено багато робіт українських й іноземних учених, а саме: С. Джонса (Jones, Bonsignour, 2012), Р. Фатрелла (Futrell, Shafer & Shafer, 2002), К. Лавріщевої (Lavrishcheva, 2013), Д. А. Маєвського (Maievskyi & Kozina, 2015), В. Яковини (Yakovyna, 2012), Г. Майєрса (Myers, Badzhett & Sandler, 2012), С. МакКоннелла (McConnell, 2013), В. Міщенка і О. Поморової (Mishhenko, Pomorova, Hovorushchenko, 2012), О. Медче (Maedche, Botzenhardt & Neer, 2012), І. Соммервілла (Sommerville, 2002), В. Харченка і Б. Конорєва (Kharchenko et al., 2012; Konorev et al., 2009), О. Харченка (Kharchenko, Galay & Yatcyshyn, 2011).

Проте, багато науковців (Matciashek, 2002; Sommerville, 2002; Kornipaev, 2013; Laplante, 2009; Wiegers, 2003) вважають, щопроцес розроблення вимог до ПЗ, особливо процедура їх формулювання, а також наявні методи і засоби забезпечення їх якості, як власне і процес розроблення самого ПЗ залишаються незабезпеченими фундаментальною теорією та ефективною методологією. Практично усі дослідження щодо розроблення вимог до ПЗ, особливо якості їх формулювання, мають хаотичний, несистематизований характер (Berenbach et al., 2009; Kroll & Krachten, 2004). Водночас, як доведено у роботах (Cockburn, 2001; Leffingwell & Widrig; Mansilla et al., 2013; Sutcliffe, 1998), саме не зовсім зрозуміле формулювання вимог до ПЗ можне призвести до появи від 35 до 55 %% всіх недоліків майбутнього програмного продукту. Безумовно, є багато фундаментальних досліджень з інженерії ПЗ (Hovorushchenko, 2018; Pomorova & Hovorushchenko, 2013; Lavrishcheva, 2013, Kharchenko & Yatsyshyn, 2009 та ін.), але відсутня завершена, протестована та апробована теорія та методологія розроблення зрозумілих і, водночас, якісних вимог до ПЗ, а також методи і засоби побудови шаблонів вимог, які можна було б застосовувати як до підготовки користувацьких, так і системних вимог. Тому проблема формулювання вимог до ПЗ потребує проведення нагальних досліджень для запобігання непередбачених втрат і неприємних інцидентів, викликаних помилками його роботи (Hrytsiuk, 2018).

Не претендуючи на кардинальні зрушення в сучасних технологіях визначення та розроблення вимог до ПЗ, спробуємо внести й свою лепту в деякі особливості формулювання вимог до ПЗ. Тому, як на сьогодні, видається нам актуальним дослідження, яке стосується розроблення методів і засобів розроблення вимог до ПЗ, особливо процедури їх формулювання з використанням відповідних шаблонів.

Об'єкт дослідження – розроблення вимог до ПЗ.

Предмет дослідження – методи та засоби розроблення вимог до ПЗ, які дадуть змогу на підставі можливостей вимог з'ясувати мовні особливості їх формулювання, встановити взаємопов'язаність та важливість вимог, уточнити деякі моменти підготовки відповідних шаблонів і їхньої деталізації.

Мета дослідження полягає у встановленні мовних особливостей формулювання вимог до ПЗ, які мають бути загальними для будь-якого процесу розроблення ПЗ, що дасть змогу на підставі їхніх можливостей з'ясувати особливості розроблення структури відповідного документа та відбору ключових вимог.

Для реалізації зазначеної мети потрібно виконати такі основні завдання:

1) проаналізувати основні можливості вимог до ПЗ та їхні атрибути, жорстко прив'язаних до цих вимог, що допоможе зрозуміти суть пропонованих принципів формулювання вимог до ПЗ;
2) з'ясувати особливості розроблення структури документа з вимогами до ПЗ та відбору ключових вимог, що дасть змогу його виконавцям здійснити чітке викладення в ньому матеріалу й завершеність думки;

3) встановити взаємопов'язаність та важливість вимог до ПЗ шляхом їх класифікації, фільтрування й сортування для того, щоб згодом отримати відносно невеликі набори вимог, кожен з яких стосується тільки однієї теми для подальшого їх аналізу;
4) виявити мовні особливості формулювання вимог до ПЗ та уточнити деякі моменти підготовки відповідних шаблонів і їхньої деталізації, які сукупно ведуть до розроблення якісного ПЗ;
5) проаналізувати критерії, які потрібно використовувати для визначення якості формулювання вимог до ПЗ, що дасть змогу аналітику дотримуватися деяких простих правил при їх написанні;
6) зробити відповідні висновки та надати рекомендації щодо практичного використання запропонованих методів і засобів формулювання вимог до ПЗ.

1. Можливості розроблених вимог до ПЗ та їхні атрибути

Перед тим, як з'ясувати особливості формулювання вимог до ПЗ та складати відповідні програмні документи, спочатку спробуємо проаналізувати можливості, які мають надавати вимоги до ПЗ, а також особливості використання атрибутів, жорстко прив'язаних до цих вимог. Це допоможе зрозуміти суть пропонованих принципів формулювання вимог до ПЗ, які детально викладено в цьому дослідженні.

Можливості, які мають надавати вимоги до ПЗ. Багато практиків вважають, що відправною точкою під час визначення вимог до ПЗ є встановлення зацікавлених сторін (англ. *Stakeholders*) так, як це показано в табл. 1. Зрозуміло, що кожна із зацікавлених сторін має знати різні можливості (корисні властивості) майбутнього ПЗ, які мають фіксувати вимоги до нього. Нижче будуть перераховані та проаналізовані всі основні дії, які можуть здійснювати всі зацікавлені сторони під час формування наборів вимог до ПЗ, а саме – їх визначення, класифікацію, оцінювання, контроль статусу та зв'язків, аналіз стосовно розділу чи всього документа, рецензування. Багато науковців і практиків вважають, що добре сформульовані вимоги до ПЗ та добре сформовані їхні набори істотно впливають на зручність роботи з ними як самих аналітиків і архітекторів, так і конструкторів його програмного коду, і відповідних тестувальників.

Табл. 1. Ролі зацікавлених сторін для формування наборів вимог до ПЗ

Зацікавлені сторони	Ролі сторін
Автор вимоги	Створює вимоги і оформляє зміни до них
Бізнес-аналітик	Формує і зберігає документ з вимогами
Експерт	Рецензує вимоги і пропонує зміни до них
Системний аналітик, конструктор коду	Аналізує вимоги і обговорює зміни до них

Вимоги до ПЗ мають надавати зацікавленим сторонам такі уявлення про їхні можливості:

- можливість однозначно ідентифікувати кожне положення вимоги;
- можливість класифікувати кожне положення вимоги такими способами:
  - за важливістю реалізації (пріоритетом виконання);
  - за типом, наприклад, функціональністю, продуктивністю, обмеженням, безпекою тощо;
  - за терміновістю (датою) реалізації – певного релізу чи версії ПЗ;
- можливість відстежувати кожну вимогу за такими статусами:
  - статусом аналізу (рецензування);
  - статусом задоволення;
  - статусом перевірки;
- можливість оцінювати вимогу з таких сторін:
  - інформації про продуктивність системи;
  - стратегії її перевірки;
  - критерію її тестування;
  - раціональності щодо реалізації;
  - наявних до неї коментарів;
- можливість аналізувати кожну вимогу стосовно певного розділу чи цілого документа, тобто в оточенні інших вимог;
- можливість знаходити в документі певні вимоги за контекстом, за класифікацією або іншими ознаками;
- можливість встановити зв'язки між вимогами й легкого переходу цими зв'язками між рівнями документа.

Використання атрибутів, жорстко прив'язаних до вимоги. З можливостей розроблених вимог до ПЗ, перерахованих вище, стає зрозумілим, що наявності простого текстового її формулювання явно недостатньо для того, щоб повністю й однозначно їх визначити. Для повного їх визначення потрібна також й інша інформація, яка їм властива, – ознаки класифікації, контроль статусу тощо.

Отже, щоб не перевантажувати зайвими деталями безпосереднє формулювання вимоги до ПЗ, фахівці-практики рекомендують виносити всю додаткову інформацію в її атрибути, жорстко прив'язані до неї. Використовуючи вміст атрибутів, системним аналітикам набагато легше обробляти користувацькі вимоги до ПЗ, здійснювати пошук потрібних вимог, впорядковувати чи формувати певні набори вимог і т.п. Набори атрибутів можна використовувати для підтримки більшості можливостей вимог, перерахованих вище, роблячи процес розроблення вимог до ПЗ більш наочним, керованим і зручним у реалізації. На рис. 1 показано приклад вимоги з властивими їй атрибутами.

[ПВ3.27] Система управління швидкою допомогою має мати здатність приймати до 100 викликів одночасно

Автор: Т. В. Мацьків, представник від замовника
Пріоритет: обов'язковий
Реліз: 1
Статус рецензування: схвалено
Можливість перевірки: так
Спосіб перевірки: моделювання, тестування

Рис. 1. Атрибути, жорстко прив'язані до вимоги

Використання того чи іншого набору атрибутів до вимоги залежить від конкретного технологічного процесу розроблення вимог до ПЗ, який підтримує ІТ-компанія. При цьому значення деяких атрибутів можна заповнювати автоматично, наприклад, встановити послідовну нумерацію або виставити дату; значення інших атрибутів має заповнювати аналітик зі слів замовника чи технічного завдання, наприклад, пріоритет реалізації або номер релізу; суть значення третіх – покажчик, який встановлюють після деякого аналізу вимог до ПЗ, наприклад, важливість, якість та спосіб перевірки.

Набір атрибутів до вимог підбирають для кожного ПЗ індивідуально, виходячи із максимальної результативності членів команди програмного проекту. Як приклад опису атрибутів до вимог К. Вігерс пропонує такий їхній набір (Wiegers, 2003):

- дата створення вимоги;

- номер поточної версії;
- автор вимоги;
- особа, що відповідає за задоволення вимоги;
- відповідальний за вимогу або перелік зацікавлених осіб (щоб прийняти рішення про запропоновані зміни);
- стан вимоги;
- походження або джерело вимоги;
- логічне обґрунтування вимоги;
- система або підсистеми, для яких призначена вимога;
- номер версії продукту проекта, для якого призначена вимога;
- метод перевірки якості наявної вимоги або критерії тестування її якості;
- пріоритет реалізації вимоги;
- стабільність вимоги.

Для розуміння зазначених атрибутів у табл. 2 наведено ті їхні категорії, які свого часу використовував англійський підрозділ INCOSE (англ. *The International Council on Systems Engineering* – Міжнародна рада з системної інженерії) під час роботи над одним з програмних проектів.

Табл. 2. Категорії атрибутів, які доцільно використовувати під час розроблення вимог до ПЗ

Категорія	Приклади значень атрибутів категорії
Ідентифікація вимоги	
− ідентифікатор	унікальний номер вимоги (ID)
− назва	унікальна коротка назва, що характеризує вимогу
Внутрішні характеристики вимоги	
− основний тип	функціональність, продуктивність системи, якість, оточення, інтерфейс, обмеження на вимогу
− якісний підтип	доступність, гнучкість, цілісність, можливість зміни, компактність, легкість підтримки та використання, кваліфікація
− тип продукту/процесу	продукт, процес, дані чи сервіс етапу реалізації проекту
− кількісний/якісний тип	кількісний, якісний продукт етапу реалізації проекту
− етап життєвого циклу	попередня чи остаточна концепція, процес розроблення, виготовлення, інтеграція/тестування, впровадження, поставка, встановлення, функціонування, підтримка, видалення, демонтаж, передавання чи утилізація
Пріоритет і важливість вимоги	
− пріоритет	ключова, необхідна, додаткова, бажана (*Key, Mandatory, Optional, Desirable*) або обов'язкова, рекомендується, можлива, доцільна (*Must, Should, Could, Wish*)
− важливість	шкала від 1 до 10
Джерело і власник вимоги	
− спосіб отримання	призначення, декомпозиція системи
− джерело	назва документа або ім'я зацікавленої особи
− власник	ім'я зацікавленої особи
− погоджено	ім'я експерта чи керівника вищого рівня
Контекст вимоги	
− набір вимог чи документів	найкраще можна управляти за допомогою правильного її внесення до структури специфікації вимог
− об'єкт, процес чи явище	що описує, відстежує чи контролює
− межі (рамки)	що має містити, а що ні, межі ПЗ
Перевірка та затвердження (*Verification* & *Validation*, або V&V) вимоги	
− V&V метод	аналіз, інспекція, системний тест, модульний тест
− V&V етап	див. етап життєвого циклу
− V&V статус	у черзі, перевірено, відхилено, не завершено
− критерій успішності перевірки	залежить від обраної декомпозиції системи
− критерій затвердження	залежить від обраного V&V методу
Підтримка процесу реалізації вимоги	
− статус узгодження	запропоновано, на узгодженні, погоджено
− статус перевірки	перевірено, не перевірено, підозріло
− статус задоволення	незадоволено, задоволено, підозріло
− статус рецензування	чекає аналізу, прийнято, відхилено
Уточнення вимоги	
− потреба	опис того, чому виникла потреба в цій вимозі
− коментарі	текстове уточнення вимоги для кращого розуміння її суті
− питання	питання, які потрібно з'ясувати для уточнення вимоги
− відповіді	відповіді, отримані при уточненні вимоги
Інші показники вимоги	
− зрілість (стабільність)	кількість змін, тривалість змін
− рівень ризику	високий, середній, низький
− оцінювана вартість	вартість, розрахована експертами
− фактична вартість	вартість, продиктована ринком
− реліз продукту	версія програмного продукту, в якій реалізована вимога

Рецензування наявних вимог до ПЗ. Не секрет, що процес розроблення більшості програмних проектів починають з його попереднього рецензування та затвердження відповідної документації. Часто наявна документація для завершених проектів слугує відправною точкою (підґрунтям) для отримання та уточнення трудовитрат на реалізацію нових аналогічних проектів. У будь-якому випадку підготовку такої документації потрібно організувати командою розробників ПЗ шляхом використання таких джерел, як поточні записи про стан реалізації програмного проекту, а також записів про всі позитивні та негативні моменти роботи над ним.

Результати формального рецензування вимог до ПЗ часто є надзвичайно корисними керівнику проекта для отримання інформації про адекватність прийнятих рішень командою розробників ПЗ під час вирішення різ-

них завдань, а також якості сформульованих вимог (рис. 2). Адже якість сформованих наборів вимог до ПЗ та їхнє формулювання є надзвичайно важливими чинниками, ніж якість всіх наступних документів, розроблених на різних етапах реалізації програмного проекту. З іншого боку, поліпшення якості вимог шляхом їх рецензування відповідними експертами дорого обходиться розробникам ПЗ. Тому керівник проекта, зазвичай, залучає літературних редакторів для того, щоб знайти тільки формальні стилістичні та граматичні помилки у формулюванні вимог, не вдаючись в деталі їхнього змісту.

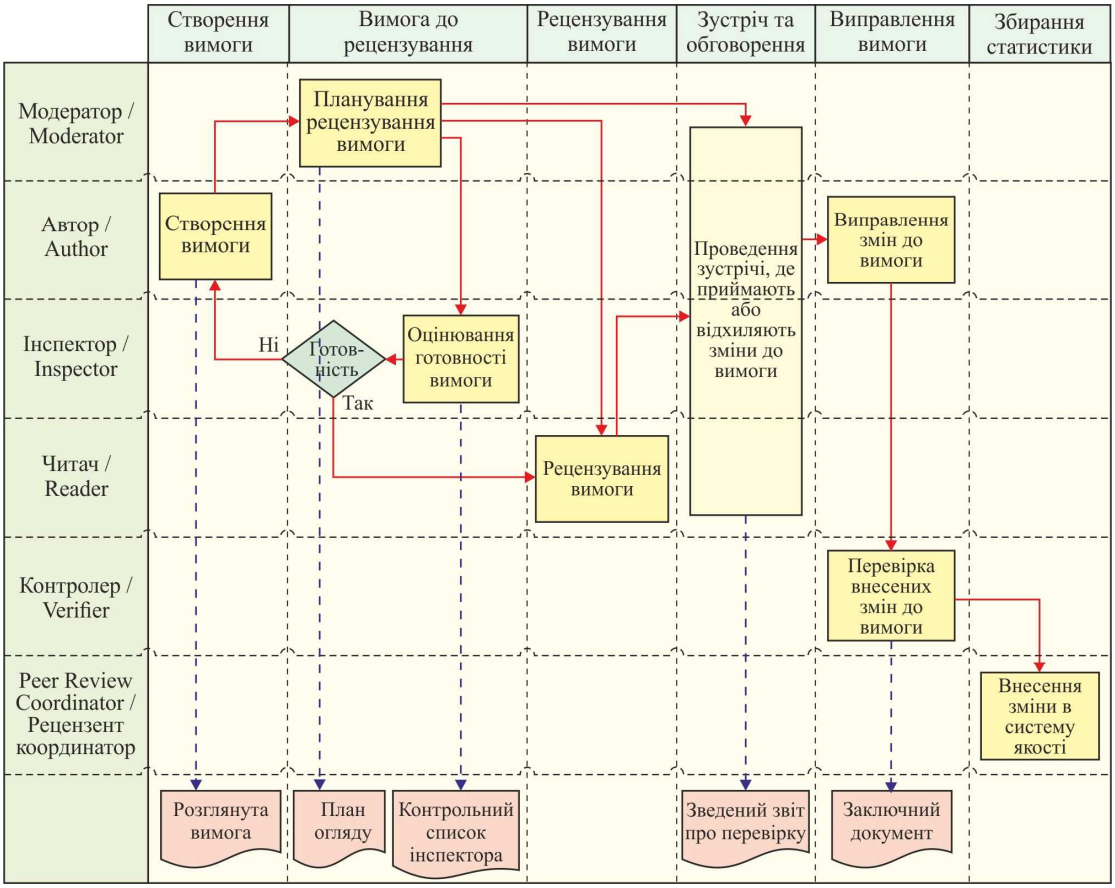

Рис. 2. Діаграма рецензування наявних вимог до ПЗ

З досвіду багатьох ІТ-компанії (Matciashek, 2002) видно, що рецензування вимог до ПЗ стає дещо ефективнішим разом із практичною їхньою перевіркою під час системного моделювання чи на конкретних продуктах проекту. Кваліфіковані рецензенти мають не просто керуватися контрольним списком з абстрактними критеріями "несуперечності, тестованості, здійсненності, ...", а мають спробувати застосувати вимоги так, щоб зробити з них, наприклад, конкретну архітектуру ПЗ, розробити відповідні тести до певного продукту проекту, запропонувати користувацьку документацію чи настанову користувача тощо. Це підвищує як якість і стабільність вимог до ПЗ, так і знижує ймовірність перероблення певних продуктів проекту на наступних етапах його реалізації, не підвищуючи при цьому загальний бюджет і терміни реалізації програмного проекту.

Відомі такі основні механізми формального рецензування вимог до ПЗ:

1. Передача документації, яка підготовлена на цей момент, кожній із зацікавлених сторін для ґрунтовної її перевірки. Після надання зацікавленим сторонам часу для самостійного рецензування вимог, потрібно зібрати їх разом для обговорення їхніх зауважень, пропозицій чи висновків. Це дасть можливість кожній зацікавленій стороні перевірити та узгодити свої міркування з усіма учасниками проекту, а також надасть можливість виявити відсутні вимоги до ПЗ.
2. Залучення кожного учасника проекту для аналізу сформованого документа з метою встановлення їхньої думки щодо його якості, виявлення будь-яких невирішених питань, додаткових чи упущених вимог або обмежень щодо можливості їх реалізації. Їхня точка зору визначається тією роллю, яку вони виконують в програмному проекті, тобто інженер з випробувань повинен мати можливість визначити тести для вимог в документації. Архітектор інфраструктури має зіставити нові функціональні можливості чи продуктивність системи для гарантованої підтримки великих додатків на обладнанні замовника або DBA (англ. *Database Administrator* – адміністратор баз даних). Також учасникам проекту необхідно визначити, чи з'явилася яка-небудь нова вимога, яка призведе до потреби збільшення фінансування та термінів реалізації програмного проекта.
3. Використання контрольних списків для перевірки кожного документа на покриття точок зору кожного учасника проекту (див. "Валідація вимог").
4. Результати рецензування вимог до ПЗ потрібно відзначити як робочі їх елементи (атрибути) або нові вимоги, або запити на зміну вимог, або завдання для подальшого аналізу вимог та їх затвердження.

2. Підготовка структури документа з вимогами до ПЗ та відбір ключових вимог

Структура будь-кого бездоганного документа, наприклад, з вимогами до ПЗ – це його внутрішня будова. Як і для будь-якого будівництва, так і для ІТ-галузі під-

готовка програмних документів має відповідати певним правилам. Головним із них є поділ документа з вимогами до ПЗ на відповідні частини. Це потрібно здійснювати для того, щоб забезпечити:

- повне викладення необхідної інформації в програмному документі. Якщо ж вона буде не систематизованою, то існує реальна небезпека того, що деяка її частина може не потрапити до нього, що спричинить недостатнє охоплення предметної області, залишить без уваги важливі факти про деякі виробничі явища та процеси тощо;
- ефективне засвоєння потрібної інформації тим, кому вона адресована – аналітикам і архітекторам ПЗ, конструкторам його програмного коду і тестувальникам, керівнику проекта та іншим зацікавленим сторонам.

Водночас, якщо початкову розрізнену інформацію завчасно не систематизувати, а згодом у програмному документі не структуризувати потрібний матеріал, то відповідному адресату доведеться докласти значних інтелектуальних зусиль для класифікації та відбору наведеної в ньому інформації. Зробити це навряд чи буде під силу кожному учаснику програмного проекта, або ж це вимагатиме від нього значних витрат часу (як це достатньо часто трапляється з так званими "робочими" записами у щоденнику керівника проекта з деякими моментами його реалізації).

Можна виділити такі основні структурні вимоги до змісту та структури програмного документа:

- зв'язність і внутрішня логіка подання матеріалу в будь-якому тексті документа зокрема та його розділах і частинах загалом;
- послідовне розміщення відповідного матеріалу в будь-якому тексті документа зокрема та його розділах і частинах загалом;
- забезпечення зручності використання документа, а також зрозумілості пошуку потрібної в ньому інформації.

Будь-який програмний документ має мати формальну визначеність, обумовлену його змістовними чинниками, логічною культурою системного мислення, рівнем формальної техніки подання матеріалу. Отже, формальна структуризація матеріалу в програмному документі дає змогу його виконавцям правильно скомпонувати зміст і структуру документа, отримати чітке викладення в ньому матеріалу й завершеність думки, логічну послідовність подання інформації, її цілісність й доступність, ясність для подальшого використання та сприйняття.

Створення оптимальної структури вимог до ПЗ. Документація з вимогами до ПЗ може займати надзвичайно великі обсяги. Наприклад, повний набір вимог до сучасної АСУ, яка має управляти навіть невеликим морським судном, може займати декілька стелажів, повністю наповнених відповідними папками. Для того, щоб доставити такий обсяг документації від замовника до виконавця, потрібно задіяти чималу вантажівку. Для уникнення такої ситуації має надзвичайно велике значення добре продумана та зрозуміла структура вимог до ПЗ, а також сама структура відповідних документів, оскільки значно полегшує їхнє розміщення та процес управління вимогами, а також відповідну реалізацію програмного проекту. Водночас, оптимальна структура вимог до ПЗ має складатися тільки з тих розділів і відповідних підрозділів, які потрібні відповідному етапу реалізації програмного проекту, що забезпечить процес розроблення ПЗ найкращої якості з мінімальними ресурсними витратами.

Створення оптимальної структури вимог до ПЗ може допомогти відповідному аналітику:

- мінімізувати загальну кількість вимог до ПЗ як користувацьких, так і системних;
- краще осмислити великий обсяг інформації, що міститься у вимогах до ПЗ;
- відшукати набори вимог до ПЗ, які стосуються певного критерію їх якості;
- виявити можливі прогалини і повторення у наявних вимогах до ПЗ;
- вилучити конфліктні ситуації (суперечності) між вимогами до ПЗ;
- управляти відповідними етапами реалізації вимог, наприклад, спочатку виконати ключові вимоги, а потім усі решта;
- відхилити малоінформативні вимоги до ПЗ після узгодження їх з автором;
- оцінити вимоги, наприклад, з позиції вартості або тривалості їх реалізації;
- повторно використовувати вимоги до ПЗ у аналогічних програмних проектах.

Зазвичай, програмний документ, що містить вимоги до ПЗ, складається з багатьох розділів і підрозділів, тобто він має свою ієрархічну структуру. Така ієрархія розділів доволі зручна для початкової класифікації вимог до ПЗ, оскільки дає змогу аналітику використовувати структуру заголовків для розподілу вимог за категоріями їх приналежності. Також ієрархічна структура подання вимог до ПЗ забезпечує їх *первинну класифікацію* через їхню позицію в загальній структурі документа. Водночас, *вторинну класифікацію вимог* можна організувати за допомогою зв'язків певних вимог з вимогами інших розділів документа або ж за допомогою цих вимог з відповідними їх атрибутами.

У роботі (Hrytsiuk, 2018, розд. 4) було описано, як часто в системному моделюванні використовують ієрархічні структури для виявлення вимог до ПЗ. Для демонстрації доцільності використання такої ієрархії можна навести такі приклади:

- декомпозиція системних цілей та можливостей користувацьких сценаріїв;
- функціональна декомпозиція системи в діаграмах потоків даних;
- декомпозиція станів системи в діаграмах станів.

У випадку, коли вимоги до ПЗ було виявлено на підставі аналізу певних системних моделей, то структуру цих моделей можна використовувати як частину структури відповідного документа. Це дасть змогу рецензентам визначити не тільки потребу вимоги у майбутньому ПЗ та якість її формулювання, але й зрозуміти їх безпосереднє походження.

Окрім конкретного формулювання вимог до ПЗ, документ, у якому зібрано ці вимоги, також може містити велику кількість різних даних, технічного, пояснювального чи уточнювального тексту, які згодом допомагають аналітикам краще зрозуміти суть кожної з вимог. До таких текстів належить:

- *супровідна інформація*, яка допомагає правильно позиціонувати вимоги стосовно розділів документа;
- *опис зовнішнього оточення* реальної системи або, як це часто називають, "знання про предметну область";
- *визначення меж дії вимоги*, які вказують, що вони мають містити, а що ні, тобто визначають межі реалізації програмного проекту;
- *визначення термінів реалізації вимоги*, тобто черговість її виконання, від якої залежить остаточний термін реалізації програмного проекту;

- *описовий текст*, який слугує для зв'язку частин розділів документа між собою та, зазвичай, причин появи чи наслідків реалізації вимог тощо;
- *характеристика зацікавленої сторони*, яка вказує на місце роботи, описує область діяльності, вирішувані завдання та наявної проблеми тощо;
- *короткий опис системної моделі*, що використовують для розроблення вимоги;
- *посилання на інші документи* – як нормативно-довідкові, так і наявні стандарти.

Формування набору ключових вимог до ПЗ. Багато ІТ-компаній починають застосовувати концепцію "ключових вимог до ПЗ" практично вже на рівні потреб користувача. Наприклад, *ключові користувацькі вимоги* KUR (англ. *Key User Requirements*) або *ключові показники продуктивності* KPI (англ. *Key Performance Indicators*) – це дві вимоги з загального їх переліку, що описують суть майбутнього ПЗ, тобто його основний функціонал.

Вибираючи ті чи інші ключові користувацькі вимоги, керівнику проекта чи аналітикам потрібно керуватися тією самою філософією, що і герої роману Джерома К. Джерома "Троє в човні, не рахуючи собаки". Вони, збираючись у тривалу подорож, дійшли до такого висновку:

> *Безперечно, Темза в своїй верхній течії недостатньо судноплавна, тому нею не зможе піднятися судно, яке вміщатиме все те, що ми вважали за необхідне взяти з собою. Джордж сказав: – "... Потрібно думати не про те, що нам може стати в нагоді, а тільки про те, без чого ми ніяк не зможемо обійтися".*

З висловлювання Джордж видно, що кожна ключова вимога має мати на увазі негативну відповідь на таке запитання користувача:

> *Якщо програмний продукт не передбачає <цю> можливість (функцію, опцію, тощо), чи в цьому випадку стану я його купувати?*

або те саме, але на системному рівні:

> *Якщо система не забезпечує <цього> виконання, то чи буде потрібна ця система все ще мені?*

У будь-якому випадку ключовими вимогами можна назвати тільки ті з них, які абсолютно потрібні чи то користувачам ПЗ, чи його розробникам.

Зрозуміло, будь-які вимоги можна аналізувати, обговорювати й коригувати, в т.ч. і ключові. Однак, варто пам'ятати, що обговорення ключових користувацьких вимог завжди потрібно проводити прискіпливо й обережно як стосовно їх формулювання, так і їх подальших змін. Там, де це доречно, кожній *ключовій користувацькій вимозі* (KUR) потрібно поставити у відповідність *ключовий показник продуктивності* (KPI). В цьому випадку замість KUR можна використовувати KPI для оцінювання ефективності пропонованих альтернативних рішень, позаяк очевидна різниця між їхніми значеннями буде переконливим аргументом відповідного вибору.

3. Взаємопов'язаність та важливість вимог до програмного забезпечення

Часто для формулювання вимог до ПЗ використовують неспеціалізовану мову, при цьому виникають деякі незручності їх інтерпретації позаяк:

- *двозначність* неспеціалізованої мови робить формулювання вимог важким для сприйняття, що часто призводить до різного їх розуміння деякими учасниками проекту;
- *гнучкість* неспеціалізованої мови, що дає змогу виражати однаковий зміст вимоги в різних формах її подання. Це може призвести до упущення суперечливих вимог, сформульованих у різних формах одних і тих самих функцій.

Формальна примітка (пояснення) до тієї чи іншої вимоги може усунути двозначність її подання, але при цьому можуть виникнути деякі питання в учасників проекту. Водночас, певні формальні пояснення можуть тільки підсумовувати чи уточнювати неспеціалізоване формулювання тієї чи іншої вимоги до ПЗ. Окрім цього, деякі важливі вимоги потрібно подавати у формі очікуваного результату, а не у формі алгоритму реалізації певних компонент ПЗ. Хоча таке формулювання є декларативним, але в деяких специфічних випадках виникає потреба викладення вимоги у формі алгоритму, особливо, якщо він є унікальним чи спеціалізованим.

Деякі важливі, наприклад, функціональні вимоги часто формулюють для того, щоб визначити зовнішні зв'язки між компонентами системи. Переважно більшість функцій системи мають бути відомі користувачам, а також іншим взаємопов'язаним системам як внутрішніми, так і зовнішніми, в т.ч. й різним сервісам мережі Інтернет. Водночас, під важливими функціями системи не потрібно розуміти тільки функції самого ПЗ. Методика декомпозиції функцій реальної системи передбачає, що найважливіші функціональні вимоги до ПЗ формулюють і поділяють за назвами її функцій. Таку методику розроблення важливих вимог до ПЗ виконують зверху-вниз.

Не секрет, що замовники ПЗ не завжди розуміють та й не цікавляться важливими функціональними можливостями програмної системи і практично ніколи не обговорюють специфічні функції майбутнього ПЗ, позаяк вони більше звертають уваги на потреби від них. Потім з таких потреб бізнес- та системні аналітики мають перетворити у вимоги до ПЗ і, що найважливіше, вони мають бути взаємопов'язані між собою. Таку методику розроблення важливих вимог до ПЗ виконують знизу-вверх.

Більшість розробників вимог до ПЗ використовують обидві методики. Спочатку бізнес-аналітик повинен розглянути загальні функції майбутнього ПЗ, а потім проаналізувати інформацію (потреби), виявлену/зібрану у майбутніх його користувачів. З цієї інформації він має згодом виокремити специфічні користувацькі вимоги, серед яких обов'язково мають бути присутні й важливі вимоги. Поділ у такий спосіб процесу розроблення вимог до ПЗ, в якому об'єднують опис функціональних можливостей системи зі специфічними функціями ПЗ, найчастіше застосовують в процесі формулювання вимог до ПЗ. При цьому враховують як взаємопов'язаність вимог між внутрішніми та зовнішніми функціями системи, так і важливість деяких вимог у їх загальній структурі.

Поняття про взаємопов'язаність вимог до ПЗ. При роботі з великими наборами вимог до ПЗ доволі часто важко ідентифікувати ті вимоги, які можуть суперечити одна іншій за змістом. Відомо (Dick, Hull & Jackson, 2017), якщо в аналітика відсутні спеціальні засоби для виявлення подібних конфліктів чи навіть незначних суперечностей, то дуже важко зрозуміти, що вимога, яка була розглянута декількома сторінками раніше від даної, має протилежний зміст. Що ж може допомогти в цьому випадку?

Відповідь на це запитання доволі проста. Потрібно мати можливість класифікувати, фільтрувати й сортувати вимоги до ПЗ для того, щоб згодом отримати відносно невеликі набори вимог, кожен з яких стосується тільки однієї теми для подальшого їх аналізу. При цьому багато вимог до ПЗ можуть одночасно стосуватися різних особливостей його функціонування. Наприклад, вимога, що стосується, переважно, питання продуктивності генератора псевдовипадкових чисел для системи захисту інформації, може зачіпати й питання інформаційної безпеки самої системи захисту. У цьому випадку таку вимогу потрібно розглядати як стосовно продуктивності генератора чисел, так і стосовно надійності системи захисту інформації.

Для підтримки названих вище можливостей вимоги до ПЗ повинні мати первинну і вторинну класифікацію. Зазвичай, кожна вимога має мати *єдину первинну класифікацію* (наприклад, її місце розташування у розділах документа) і *множину вторинних класифікаційних властивостей*, що використовують можливості атрибутів вимог і їхніх зв'язків. Така методика подання вимог до ПЗ істотно допомагає аналітику їх аналізувати, а відповідним експертом – рецензувати вимоги. Це дає їм змогу знаходити всі вимоги, що за змістом пов'язані між собою, а також за допомогою процедур фільтрування та впорядкування за ключовими словами використовувати ознаки основної та додаткової класифікацій.

Наприклад, щоб звузити поле можливого пошуку таких вимог, спочатку потрібно сформувати набір всіх вимог до ПЗ, що стосуються надійності системи захисту інформації. А вже потім серед відібраних вимог потрібно проаналізувати схожі вимоги на предмет наявності конфліктів між ними.

Поняття про важливість вимог до ПЗ. Деякі вимоги до ПЗ належить до категорії так званих "не обговорюваних" вимог, тобто їх зміст чи призначення не варто аналізувати, а потрібно відразу ж виконувати. Бо, якщо готовий програмний продукт не задовольнятиме такі вимоги, то його просто не будуть ефективно використовувати безпосередні користувачі. Відповідно, всі інші вимоги до ПЗ можна як обговорювати й аналізувати, так і коригувати. Наприклад, якщо, згідно з вимогою, система управління роботою швидкої допомоги має забезпечувати одночасну роботу як мінімум зі 100 абонентами, а готове рішення підтримує тільки 99 абонентів, то таке рішення, найімовірніше, буде все-таки визнано задовільним результатом і корисним для замовника, і повчальним для його розробника.

Процес оцінювання ступеня важливості (задоволеності) вимоги до ПЗ є складним і важким завданням. Можливо, для попереднього прикладу досягнення показника одночасної роботи з 95 абонентами є ще прийнятною (але незадовільною) величиною для системи управління роботою швидкої допомоги, а ось будь-яка величина, менша за 90 абонентів, буде вже категорично неприйнятна. Водночас, показник у 110 абонентів швидше за все є відмінним результатом, який для замовника буде навіть набагато ціннішим, ніж 100 абонентів.

Одним з підходів, що полегшує вирішення цієї проблеми, є визначення декількох значень продуктивності системи для одного показника, наприклад:

- **О** (обов'язковий) – обов'язкова верхня (або нижня) межа значення величини;
- **Б** (бажаний) – бажане значення;
- **Н** (найкращий) – оптимальне значення.

Кожне з цих значень можна зберігати у власному атрибуті вимоги до ПЗ або ж їх можна описати безпосередньо в тексті вимоги, наприклад, у такій формі:

"*Система має забезпечувати одночасну роботу з* [**О**: 90, **Б**: 110, **Н**: 100] *абонентами*".

Дещо інший підхід полягає в тому, щоб графічно (за допомогою певної функції) відобразити значення важливості вимоги до ПЗ залежно від значення показника продуктивності системи. В цьому випадку важливість вимоги, зазвичай, знаходиться в межах від 1 до 100 одиниць. Для розуміння цього, на рис. 3 показано приклади, які відображають різну форму функцій важливості вимоги до ПЗ:

1) функція (*а*) демонструє випадок, коли замовнику бажано, щоб кількість абонентів, з якими одночасно може працювати система, прямувала до максимуму, але при цьому визначено й мінімально допустиме значення показника продуктивності системи – 90 абонентів;
2) функція (*б*) демонструє бінарний випадок – або певна продуктивність системи (в 100 абонентів) можна досягти, або ні. При цьому навіть 120 абонентів, з якими одночасно може працювати система, не приносять жодної додаткової користі її замовнику;
3) функція (*в*) відображає випадок, коли значення показника продуктивності системи потрібно мінімізувати (наприклад, вага пристрою), але при цьому визначена й максимально допустима вага – 20 кг;
4) функція (*г*) відображає випадок, коли значення показника продуктивності системи потрібно оптимізувати (наприклад, кількість обертів двигуна – 20 об/хв).

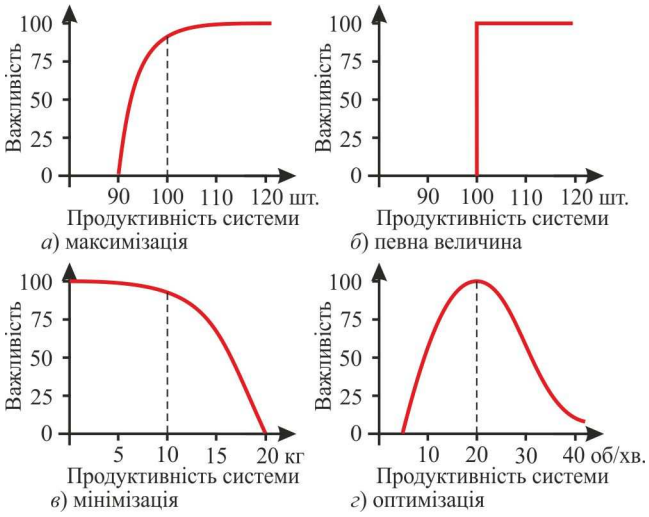

Рис. 3. Типові функції важливості вимог до ПЗ

Використання графічних функцій є доволі наочним способом подання важливості вимог до ПЗ. Один погляд на функцію дає аналітикам уявлення про суть вимоги – потрібно мінімізувати, максимізувати, оптимізувати значення показника продуктивності системи чи забезпечити його деяке конкретне значення.

Такий підхід до визначення важливості вимог до ПЗ має додаткову перевагу над іншими способами, позаяк дає змогу системним аналітикам усвідомити ступінь їхньої свободи при виробленні та прийнятті правильного рішення. Це означає, що для окремих вимог до ПЗ шляхом узгодження значень певних показників продуктивності системи можна отримати найкращі їх значення загалом. Цей підхід дуже часто використовують при

проведенні тендерів для порівняння та оцінювання однотипних варіантів альтернативних пропозицій.

Для відображення значення вимоги до ПЗ можна використовувати також і атрибут, що містить конкретне значення функції залежно від продуктивності системи, особливості реалізації якого розглянемо нижче.

4. Мовні особливості формулювання вимог до ПЗ, підготовка їх шаблонів і деталізація

Тут спробуємо описати детально мовні особливості формулювання вимог до ПЗ, а також висвітлимо деякі особливості підготовки шаблонів вимог і підходи до їх деталізації, які сукупно ведуть до розроблення якісного ПЗ. При цьому матимемо на увазі, що замовники ПЗ не завжди можуть розуміти деякі технічні моменти виконання таких процедур. Тим не менше, якісне формулювання вимог – тісна співпраця бізнес-аналітика з замовником ПЗ, в якому замовник і розробник встановлюють і записують ряд відповідних домовленостей, які ведуть до узгодження відповідної специфікації вимог до ПЗ.

Зазвичай, в ІТ-компанії бізнес-аналітики мають мати прямий контакт з його представниками. Постійне обговорення з представниками замовника, які здебільшого будуть користувачами майбутнього ПЗ, призводить до формулювання всіх технічних деталей його роботи, які стосуються їх безпосередньої діяльності. При цьому прийнято, щоб одна кваліфікована особа з боку замовника була відповідальна за комунікацію з бізнес-аналітиком від розробника ПЗ.

Труднощі на цьому етапі виникають з таких причин:

- замовник ПЗ не завжди знає, які його майбутні цілі чи як їх можна досягнути, адже існує багато способів їх досягнення;
- великі програмні системи можуть використовувати багато користувачів. Вони можуть мати різні підходи до їх використання, а також можуть володіти різною термінологію. Уніфікація як термінології, так і шаблонне подання їхніх потреб – одна із основних цілей формулювання якісних вимог до ПЗ;
- особи, що організовують замовлення і безпосередні користувачі – це різні люди. Думка замовника може бути критичною на початковому етапі визначення вимог до ПЗ, але вони, можливо через недостатність знань чи відсутність певної інформації, не враховують деяких моментів роботи майбутнього ПЗ та, що найважливіше, майбутніх ризиків реалізації програмного проекту.

Використання чіткої та зрозумілої мови для формулювання вимог до ПЗ. Використання чіткої та зрозумілої мови (узгоджених термінів) для формулювання вимог до ПЗ дає змогу аналітикам здійснити їх класифікацію чи структуризацію, що істотно полегшує подальше їх розуміння розробниками ПЗ. Простим прикладом тут може слугувати використання в тексті слова "має" та "повинна" як ключового слова, що означає звичайну наявність вимоги – у першому випадку, інакше – обов'язковість її виконання. Деякі методики формулювання вимог до ПЗ навіть заставляють використовувати такі ключові слова, які вирізняються між собою, для встановлення пріоритету вимог, наприклад, – "має" (*must*), "рекомендується" (*should*) і "можливо" (*may*).

Багато практиків вважають, що мова, яку використовують для формулювання вимог до ПЗ, значною мірою залежить від рівня підготовки документа з вимогами. Йдеться про те, що існує принципова відмінність між користувацькими вимогами, які належать до області наявних проблем, і системними вимогами, які належать до області прийняття рішень (Hrytsiuk, 2018, розд. 2.7).

Як зазначалося в (Hrytsiuk, 2018, розд. 4), користувацькі вимоги, переважно, описують можливості майбутнього ПЗ (послуги, які має надавати ПЗ), необхідні користувачам для їх повсякденної роботи, або обмеження, які дещо скорочують ці можливості. В цьому випадку вимога, яка стосується такої можливості, має описувати тільки одну потребу, необхідну або для одного користувача, або групи з декількох однотипних користувачів. При цьому в тексті вимоги потрібно обов'язково вказувати тип такого користувача.

Типова вимога, яка описує можливість користувача, виглядає так:

<Тип користувача> *повинен мати можливість* ***<опис можливості>***

Наприклад, загальна вимога до потреби користувача та продуктивності системи може мати такий вигляд:

Оператор *повинен мати можливість* ***зробити постріл.***

Якщо існують певні вимоги до продуктивності системи або обмеження, пов'язані тільки з однією конкретною вимогою, то її формулювання в цьому випадку можна доповнити, після чого вона виглядатиме вже так:

<Тип користувача> *повинен мати можливість* ***<опис можливості>*** *протягом* ***<показник продуктивності системи>*** *від* ***<моменту відліку>****, перебуваючи в* ***<умови використання системи>***

Наприклад, сформульована вище загальна вимога в окремому випадку може містити умови продуктивності системи й обмеження та виглядати так:

Оператор *повинен мати можливість* ***зробити постріл*** *протягом* ***3 секунд*** *від моменту* ***виявлення цілі радаром****, перебуваючи в* ***складних морських умовах.***

Набагато рідше трапляється ситуація, коли атрибут з однією і тією самою умовою продуктивності системи пов'язаний з декількома різними вимогами. Можна уявити ситуацію, коли декілька різних вимог характеризує один і той же тимчасовий параметр.

Однак, на практиці, коли існує ієрархія вимог і вимоги більш низького рівня є деталізацією вимог вищого рівня, то це найчастіше означає, що необхідне значення атрибута, наприклад, продуктивності системи фактично пов'язане з вимогою дещо вищого рівня. Як відомо (Dick, Hull & Jackson, 2017), всі вимоги нижчого рівня, які є деталізацією вимог вищого рівня, просто наслідують значення їхніх атрибутів.

Часто можна бачити, що обмеження до можливостей системи описують не в тексті самої вимоги, а окремо. Пов'язано це з тим, що таке обмеження або стосується повністю всієї системи й немає сенсу багато разів повторювати його в кожній вимозі, або навпаки, це обмеження стосується різних особливостей процесу розроблення ПЗ, тому його потрібно виділяти окремо для подальшого контролю.

Зазвичай, обмеження в користувацьких вимогах до ПЗ згадуються або як мінімально прийнятні параметри продуктивності системи, що заявляються замовником, або є наслідком потреби взаємодії системи з оточенням (зазвичай, сюди належать правові та соціальні системи).

Вимогу типу обмеження, зазвичай, виражають в такій формі:

<Тип користувача> *не повинен потрапляти під дію* ***<відповідне законодавство>****, що передбачає відповідальність за* ***<тип порушення>***

Приклад з реального життя:

Водій швидкої допомоги *не повинен потрапляти під дію* ***законодавства****, що передбачає відповідальність за* ***порушення правил дорожнього руху****.*

Також у вимозі можна вводити й інші типи обмежень, наприклад:

<Тип користувача> *не повинен потрапляти під дію* ***<відповідне законодавство>****, що передбачає відповідальність за* ***<тип порушення>****, за умови* ***<тип обмеження>*** *і* ***<тип порушення>***

Той самий приклад з реального життя матиме вже такий вигляд:

Водій швидкої допомоги *не повинен потрапляти під дію* ***законодавства****, що передбачає відповідальність за* ***порушення правил дорожнього руху****, за умови* ***увімкнутої сирени*** *і* ***увімкнутих проблискових маяків****.*

Оскільки обмеження стосуються області прийняття рішень, то мова системних вимог до ПЗ є відмінною від мови користувацьких вимог. При формулюванні системних вимог до ПЗ системні аналітики мають зосереджувати свою увагу переважно на описі системної функціональності ПЗ та на побудові самих обмежень. Конкретне формулювання системної вимоги залежить також від типу обмеження або значення показника продуктивності системи, які пов'язані з цією вимогою.

Наведемо приклад загального опису функціональності будь-якого ПЗ, яка містить необхідне значення показника продуктивності системи, наприклад, навантаження на неї:

<Система> *повинна* ***<виконувана функція>*** *не менше, ніж* ***<кількість> <об'єкт>****, функціонуючи в* ***<умови використання системи>****.*

Або в конкретних реаліях:

Телекомунікаційна система *повинна* ***забезпечувати телефонний зв'язок*** *не менше, ніж* ***з 10 абонентами****, функціонуючи в* ***умовах відсутності зовнішнього джерела електроживлення****.*

Наведемо ще один приклад, який описує періодичне обмеження:

<Система> *повинна* ***<виконувана функція> <об'єкт>*** *кожні* ***<показник продуктивності системи> <одиниця виміру>***

Мовою настанови це виглядає так:

Кава-машина *повинна* ***виготовляти гарячий напій*** *кожні* ***10 секунд.***

Обговорення цієї теми буде продовжено в наступних розділах.

Підготовка шаблонів вимог до ПЗ та обмежень до них. У попередньому розділі було продемонстровано особливість використання чіткої та зрозумілої мови (узгоджених термінів) для формулювання вимог до ПЗ за допомогою деяких шаблонів. Тут продовжимо висвітлення цієї теми стосовно підготовки та формулювання вимог з використанням різних шаблонів.

Використання шаблонів, як це було показано на прикладах у попередньому розділі, є хорошим способом стандартизації мовних викладок, як застосовують для формулювання вимог до ПЗ. Для того, щоб мати можливість стандартно писати вимоги різних типів, потрібно просто зібрати перелік таких шаблонів. У ході практичного застосування набору таких шаблонів їх можна уточнювати, розширювати й коригувати. Це потрібно робити для того, щоб у підсумку отримати дещо повніший за структурою і мовно досконалий набір уніфікованих шаблонів, який надалі можна навіть використовувати й в інших програмних проектах.

Отже, процедуру формулювання вимог до ПЗ за допомогою шаблонів поділимо на два етапи:

- вибір найбільш придатного шаблону із загального набору шаблонів;
- підстановка конкретних даних для заповнення порожніх полів у шаблоні.

Такий підхід дає можливість аналітику розділити шаблон і дані, які потрібно підставляти до нього. Тоді будь-яка вимога просто міститиме посилання на шаблон, а дані можна фіксувати окремо – як атрибути цієї вимоги. На рис. 4 наведено приклад, який демонструє такий "шаблонний" підхід, який дає можливість аналітику в будь-який потрібний момент згенерувати текстове формулювання вимоги до ПЗ.

Використання уніфікованих шаблонів для формулювання вимоги до ПЗ має такі переваги:

- *можливість глобальної зміни стилю подання вимоги:* для зміни формулювання визначеної вимоги потрібно внести зміну тільки в один або декілька конкретних шаблонів, які задіяні в цій схемі;
- *можливість дещо легшого оброблення інформації*: наприклад, виділення всіх полів, що стосуються <умов використання системи>, в окремий атрибут вимог дає змогу дещо зручніше фільтрувати й сортувати вимоги, враховуючи конкретні ознаки умови використання системи;
- *можливість захисту конфіденційної інформації*: в тому випадку, коли вимоги містять конфіденційну або секретну інформацію, шаблони можна використовувати для захисту саме тієї частини тексту вимоги, доступ до якої має бути захищений.

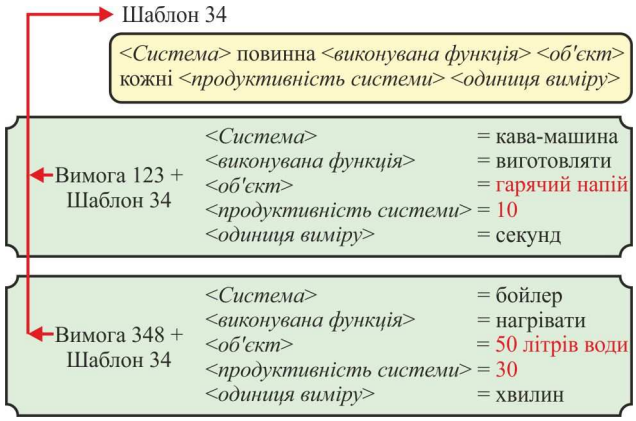

Рис. 4. Уніфіковані шаблони вимог до ПЗ

Останній пункт потребує деякого уточнення. В оборонних (військових) і деяких комерційних (наприклад, банківських) проектах потрібно обмежувати доступ деяких учасників проекту, але не до всієї інформації, а тільки до певної її частини. Дуже часто текст однієї вимоги може містити інформацію, що стосується різних рівнів секретності.

Наприклад, повністю очевидно (не таємно), що сучасний військовий корабель буде оснащений ракетами, які він за потреби має запускати у певну ціль, однак така інформація про продуктивність системи, зазвичай, має закритий характер: можлива кількість запусків у одиницю часу, радіус та цілі ураження та ін.

Замість того, щоб обмежувати доступ деяких учасників проекту до цілої вимоги тільки через те, що деяка її частина є конфіденційною, цей підхід дає можливість

переглядати всі вимоги усіма зацікавленими сторонами, але без доступу до конфіденційної інформації, що міститься в деяких її атрибутах. Насправді, за такого підходу різні учасники проекту (з різним рівнем доступу) можуть бачити тільки різні набори атрибутів для однієї і тієї ж вимоги до ПЗ.

Використання обмежень у шаблонах вимог. Однією з найскладніших особливостей процесу формулювання вимог до ПЗ та одним з найбільш традиційних типів вимог є так звані *шаблони вимог з обмеженнями*. Саме врахування обмежень у шаблонах вимог істотно полегшує якісне їх формулювання. Однак, тут необхідно також врахувати один передбачуваний недолік – для української мови застосування цього методу може ускладнюватися потребою узгодження відмінків.

Отже, відомий такий підхід до підготовки шаблонів вимог з обмеженнями:

1) Зібрати всі можливі вимоги до ПЗ, у яких використовують певні обмеження.
2) Підготувати набір обмежень різних типів, які можуть траплятися в ході реалізації програмного проекту. Якщо цей перелік ґрунтується на попередньому досвіді реалізації аналогічного проекту, то маючи деякий готовий набір шаблонів з обмеженнями, його можна використовувати як основу для опису обмежень поточного проекту. Інакше доведеться розробляти заново шаблони вимог до ПЗ.
3) Проаналізувати для кожної вимоги повний перелік можливих обмежень з підготовленого набору й визначити, які саме з них можна застосовувати для певної вимоги. Тут надзвичайно зручно використовувати таблицю, де стовпці відповідають різним типам обмежень, а рядки – їхнім формулюванням. Якщо для вимоги потрібно додати обмеження певного типу, то у відповідній клітинці потрібно сформулювати це обмеження; якщо ж потреби в обмеженні взагалі немає, то у відповідній клітинці потрібно поставити "немає" (або N/A, *Not Available*).
4) Вибрати для кожного обмеження найпридатніший для нього шаблон і сформулювати вимогу за допомогою нього.
5) Завершити процес тоді, коли всі елементи таблиці заповнені.

Використання такого підходу для підготовки шаблонів вимог до ПЗ з обмеженнями дає змогу аналогічно відповісти на два традиційні запитання:

- Як формулювати вимогу з обмеженням? *Відповідь: потрібно використовувати шаблони.*
- Як переконатися в тому, що всі обмеження враховані? *Відповідь: потрібно використовувати таблицю для аналізу покриття вимог з відповідними обмеженнями.*

У табл. 3 показано приклад набору шаблонів для формулювання вимог до ПЗ з обмеженнями. Здебільшого, для одного типу обмежень можна використовувати різні шаблони, а самі обмеження можуть мати складну класифікацію. У наведених прикладах тільки текст, виділений жирним шрифтом, стосується безпосередньо самого обмеження.

Табл. 3. Приклади формулювання шаблонів вимог до ПЗ з обмеженнями

Тип обмеження	Формулювання шаблону вимог до ПЗ
Продуктивність чи можливість системи	***<Система>*** *повинна* ***<виконувана функція> <об'єкт>*** *не менше, ніж* ***<продуктивність системи>*** *разів у* ***<одиниця виміру>***
Продуктивність чи можливість системи	***<Система>*** *повинна* ***<виконувана функція> <об'єкт>*** *типу* ***<характеристика>*** *протягом* ***<можливість системи> <одиниця виміру>***
Продуктивність чи потужність системи	***<Система>*** *повинна* ***<виконувана функція>*** *не менше, ніж* ***<кількість> <об'єкт>***
Продуктивність чи своєчасність системи	***<Система>*** *повинна* ***<виконувана функція> <об'єкт>*** *протягом* ***<своєчасність системи> <одиниця виміру>*** *від моменту* ***<подія>***
Продуктивність чи періодичність системи	***<Система>*** *повинна* ***<виконувана функція>*** *не менше, ніж* ***<кількість> <об'єкт>*** *протягом* ***<продуктивність системи> <одиниця виміру>***
Здатність до взаємодії чи потужності системи	***<Система>*** *повинна* ***<виконувана функція> <об'єкт>*** *складається з не менше, ніж* ***<потужність системи> <одиниця виміру>*** *з* ***<зовнішня сутність>***
Стійкість чи періодичність системи	***<Система>*** *повинна* ***<виконувана функція> <об'єкт>*** *з* ***<стійкість системи> <одиниця виміру>*** *кожні* ***<стійкість системи> <одиниця виміру>***
Оточення чи роботоздатність системи	***<Система>*** *повинна* ***<виконувана функція> <об'єкт>****, функціонуючи в* ***<умови використання системи>***

Особливості деталізації вимог до ПЗ. Використання шаблонів для формулювання вимог до ПЗ дає змогу аналітикам звернути увагу на те, що деякі обмеження або показники продуктивності системи можна окремо сформулювати, тобто можуть існувати як окремі підпункти (підкласи) відповідних функціональних вимог. Це дає змогу виділяти ці підкласи у вигляді окремих вимог та встановлювати для них зв'язки з відповідними функціональними вимогами.

Однак, у допитливого читача може виникнути питання про ступінь (глибину) деталізації інформації. А саме, як саме ми збираємося "розщеплювати атом" під час розроблення вимог до ПЗ? На це запитання можна відповісти так: вимогу можна дробити на підпункти доти, доки засіб підтримки роботи з вимогами (англ. *Requirements Management Tool*) дає можливість аналітику бачити кожну вимогу в потрібному контексті.

На рис. 5 наведено спосіб поділу вимоги, за якого підпункти стають дочірніми елементами до основної вимоги, утворюючи в такий спосіб певну ієрархію. При цьому головна вимога до ПЗ залишається придатною для читання та розуміння сама по собі, водночас, як дочірні її елементи можна розглядати тільки стосовно "батьківської" вимоги, хоча при відстеженні та встановленні зв'язків вони можуть існувати повністю автономно.

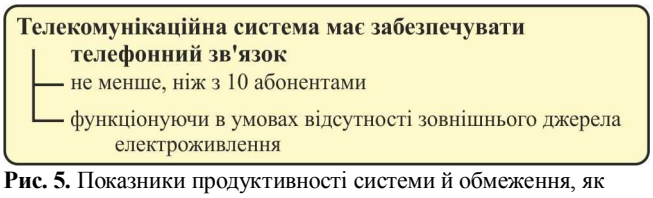

Рис. 5. Показники продуктивності системи й обмеження, як підпункти

Це означає, що ми можемо працювати з дочірнім елементом як з повністю завершеною вимогою, придатною для відстеження, але цитувати (пов'язувати) дочірні елементи краще все-таки разом із текстом батьківської вимоги до ПЗ.

Якщо знову звернутися до прикладу, наведеного на рис. 4, виділивши курсивом текст батьківської вимоги в тексті дочірніх елементів, то отримаємо таке:

- телекомунікаційна система має забезпечувати телефонний зв'язок;
- *телекомунікаційна система має забезпечувати телефонний зв'язок* не менше, ніж з 10 абонентами;
- *телекомунікаційна система має забезпечувати телефонний зв'язок,* функціонуючи в умовах відсутності зовнішнього джерела електроживлення.

Існують різні способи організації ієрархії таких підкласів. Припустимо, що потрібно описати декілька різних можливостей, які мають бути доступні вимозі "функціонуючи в умовах відсутності зовнішнього джерела електроенергії". Варіант подання вимоги до ПЗ з її деталями подано на рис. 6.

Для цього прикладу взаємозв'язок між вимогами до ПЗ виглядатиме дещо інакше:

- *функціонуючи в умовах відсутності зовнішнього джерела електроенергії*, телекомунікаційна система має забезпечувати телефонний зв'язок;
- *функціонуючи в умовах відсутності зовнішнього джерела електроенергії, телекомунікаційна система має забезпечувати телефонний зв'язок* не менше, ніж з 10 абонентами;
- *функціонуючи в умовах відсутності зовнішнього джерела електроенергії, телекомунікаційна система* має забезпечувати радіозв'язок;
- *функціонуючи в умовах відсутності зовнішнього джерела електроенергії, телекомунікаційна система має забезпечувати радіозв'язок* не менше, ніж з 15 водіями швидкої допомоги.

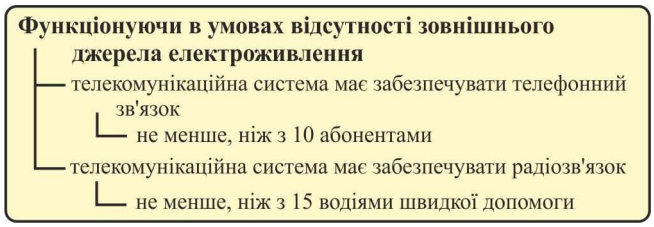

Рис. 6. Альтернативне подання підпунктів вимоги до ПЗ з її деталями

Це означає, що суть такого способу деталізації вимог до ПЗ полягає в тому, що під час побудови ієрархії вимог до ПЗ (їх деталізації) батьківську вимогу потрібно формулювати так, щоб забезпечувати повний смисловий контекст для кожного дочірнього елемента, в т.ч. і можливу подальшу деталізацію дочірніх елементів на дещо дрібніші блоки інформації.

5. Критерії, яким має відповідати зміст сформульованих вимог до ПЗ

Незалежно від мовних особливостей процесу формулювання вимог до ПЗ, існують чіткі критерії, яким мають задовольняти зміст і якість сформульованої вимоги. До цих критеріїв належать такі:

- *атомарність*: кожне твердження (формулювання вимоги) має виглядати як один елемент ієрархії, придатний для встановлення зв'язків з ним;
- *унікальність*: кожна вимога має мати власну унікальну ідентифікацію;
- *здійсненність*: вимоги можна технічно реалізувати в установлені терміни і в межах виділеного бюджету;
- *законність*: кожна вимога немає суперечити чинному законодавству;
- з*розумілість*: кожна вимога має бути дохідливо сформульована, має вилучати неоднозначне тлумачення;
- *точність*: кожна вимога має бути точною та лаконічною;
- *можливість перевірки*: для кожної конкретної вимоги має існувати прозорість її реалізації;
- *абстрактність*: формулювання вимоги немає нав'язувати певні технічні рішення, характерні для нижчих рівнів похідних вимог.

Додатково існують певні критерії, які можна застосовувати для формування наборів вимог до ПЗ:

- *повнота*: всі вимоги мають бути зафіксовані та згруповані у певні набори;
- *несуперечність*: немає існувати вимог, що суперечать одна іншій;
- *відсутність надмірності*: кожна вимога має бути сформульована тільки один раз (без дублювання), в якій немає уточнюватися повторно одна і та сама інформація;
- *модульність*: вимоги, близькі одна до іншої за змістом, мають міститися в одному наборі вимог;
- *структурованість*: наявність зрозумілої та чіткої структури документа для узгодження та затвердження вимог;
- *задоволеність*: досягнутий необхідний рівень покриття вимог зв'язками типу "задовольняється / задовольняє";
- *тестованість*: досягнутий необхідний рівень покриття вимог тестами.

Для демонстрації того, як ***не потрібно робити***, нижче наведено два відомі приклади формулювання вимог згідно з розглянутими вище критеріями:

1) *Система має забезпечувати максимальний рівень її продуктивності протягом всієї тривалості роботи, за винятком аварійних ситуацій, коли вона має забезпечувати рівень продуктивності до 125%, але тільки тоді, якщо аварійна ситуація триває не більше, ніж 15 хв, – в іншому випадку система має зменшити рівень продуктивності до 105%; але в разі, якщо вдається досягти рівня продуктивності тільки 95%, система має активувати режим "винятково малого рівня" і підтримувати цей рівень у межах 10% від початкового значення протягом 30 хв як мінімум.*
2) *Система має забезпечувати основні функції текстового редактора, зручні для використання ненавченим персоналом, і має працювати в умовах "тонкого" Ethernet'а, прокладеного повітряною системою кабельних каналів з інтегрованими адаптерами, що поставляються з додатковими модулями пам'яті, за потреби.*

Ці приклади демонструють *класичні негативні ситуації*, характерні для розробників вимог до ПЗ. Для того, щоб уникнути цих ситуацій, аналітику потрібно дотримуватися таких простих правил:

- *уникати хаосу*: формулюючи вимоги до ПЗ, потрібно сконцентрувати на найважливішому; вимога не має бути схожою на роман;
- *уникати "лазівок"*: наприклад, таких виразів, як "якщо це потрібно", оскільки такі "дірки" роблять вимогу неоднозначною і часто марною;
- *уникати розміщення понад однієї вимоги в одному розділі*: часто наявність у одному пункті понад однієї вимоги легко визначити за наявності сполучника "і";
- *уникати міркувань:* здебільшого, роздуми мають свою певну схему і логічну структуру. Спочатку визначають головну думку, формують гіпотезу або тезу. Потім, щоб її підтвердити або спростувати, наведено доведення та аргументи. У підсумку кінцівка тексту має містити логічно обґрунтовані висновки;
- *уникати використання "розмитих" понять і слів*: природно, здебільшого, зазвичай, переважно, часто, інколи, нормально, типово, на загал;
- *уникати використання невизначених термінів:* наприклад, зручний у використанні, універсальний, гнучкий, очевидний;

- *уникати прийняття бажаного за необхідне*: наприклад, 100% надійний, приємний для всіх користувачів, безпечний у використанні, відповідний для всіх платформ, не мав би ніколи ламатися, мав би обробляти всі несподівані збої системи та бути готовим до модернізацій для будь-яких ситуацій, які можуть виникнути в майбутньому і т.д.

Аналіз першого прикладу показує, що замість однієї вимоги потрібно писати 12. Розвиваючи цю думку, найкраще виділити 4 окремих умови використання системи – нормальне, аварійне, аварійне понад 15 хв, режим "винятково малого рівня", – і описати окремі вимоги для кожного з цих умов. Варто звернути увагу на наявну лазівку в другому прикладі. Абсолютно незрозумілий обсяг вимоги. Наприклад, цю вимогу можна інтерпретувати й так: "*Система має забезпечувати основні функції текстового редактора ... за потреби*".

Однак, аналітикам-початківцям потрібно пам'ятати, що не існує досконалого та вивіреного способу написання ідеальних вимог, а кращий вчитель – це досвід, який вони напрацьовують протягом тривалого часу. Бездоганну документацію з якісними формулюваннями вимог до ПЗ відрізняє технічний стиль їх викладення та користувацька термінологія, а не комп'ютерний сленг. Водночас, як би не виглядав технологічний процес розроблення ПЗ, яким би змінам він не піддавався, певна методика структуризації вимог до ПЗ (як користувацьких, так і системних), конкретні мовні особливості їх формулювання та відповідне рецензування мають залишатися незмінними протягом усіх етапів реалізації програмного проекту.

Висновки

Встановлено мовні особливості процедури формулювання вимог до ПЗ, які є загальними для будь-якого процесу його розроблення, що дає змогу аналітику на підставі їхніх можливостей з'ясувати особливості розроблення структури відповідного документа та здійснити відбір ключових вимог з множини допустимих. За результатами дослідження можна зробити такі основні висновки.

1. Проаналізовано основні можливості вимог до ПЗ та їхні атрибути, жорстко прив'язані до цих вимог, що допомогло зрозуміти суть пропонованих принципів формулювання вимог до ПЗ.

2. З'ясовано особливості розроблення структури документа з вимогами до ПЗ та відбору ключових вимог, що дає змогу його виконавцям здійснити чітке викладення в ньому матеріалу й завершеність думки, логічну послідовність подання інформації, її цілісність й доступність, ясність для подальшого використання та сприйняття.

3. Встановлено взаємопов'язаність та важливість вимог до ПЗ шляхом їх класифікації, фільтрування й сортування для того, щоб згодом отримати відносно невеликі набори вимог, кожен з яких стосується тільки однієї теми для подальшого їх аналізу.

4. Виявлено мовні особливості процедури формулювання вимог до ПЗ та уточнено деякі моменти підготовки відповідних шаблонів і їхньої деталізації, які сукупно ведуть до розроблення якісного ПЗ шляхом тісної співпраці бізнес-аналітика з його замовником, де вони встановлюють і записують відповідні домовленості.

5. Проаналізовано критерії, які потрібно використовувати для визначення якості формулювання вимог до ПЗ, що дає змогу аналітику дотримуватися деяких простих правил при їх написанні.

6. Зроблено відповідні висновки та надано рекомендації щодо практичного використання запропонованих методів і засобів формулювання якісних вимог до ПЗ.

Перелік використаних джерел

Berenbach, B., Paulish, D., Katzmeier, J., & Rudorfer, A. (2009). Software & Systems Requirements Engineering: In Practice. New York: McGraw-Hill Professional.

Dick, J., Hull, E., & Jackson, K. (2017). *Requirements Engineering*. (4rd ed.) Springer. https://doi.org/10.1007/978-3-319-61073-3

Futrell, R. T., Shafer, D. F., & Shafer, L. I. (2002). *Quality Software Project Management*. Prentice Hall. 1639 p.

Hovorushchenko, T. O. (2018). Teoretychni ta prykladni zasady informatsiinoi tekhnolohii otsiniuvannia dostatnosti informatsii shchodo yakosti u spetsyfikatsiiakh vymoh do prohramnoho zabezpechennia. *Abstract of Doctoral Dissertation for Technical Sciences* (05.13.06 – Information technologies). Lviv: Ukrainska akademiia drukarstva. 43 p. [In Ukrainian].

Hrytsiuk, Yu. I. (2018). *Analysis of Software Requirements: Tutorial. Lviv: Publishing House of Lviv Polytechnic*. 460 p. [In Ukrainian].

Hrytsiuk, Yu. I., & Leshkevych, I. F. (2017). The Problems of Definition and Analysis of Software Requirements. *Scientific Bulletin of UNFU, 27*(4), 148–158. https://doi.org/10.15421/40270433

Jones, C., & Bonsignour, O. (2012). *The economics of software quality*. Boston: Pearson Education. 588 p.

Kharchenko, A., Galay, I., & Yatcyshyn, V. (2011). The method of quality management software. *VII International Conference on Perspective Technologies and Methods in MEMS Design: Proceedings*, (pp. 82-84), May 11-14, 2011. Polyana (Ukraine).

Kharchenko, O., & Yatsyshyn, V. (2009). Rozrobka ta keruvannia vymohamy do prohramnoho zabezpechennia z vy-korystanniam modeli yakosti PZ. *Visnyk Ternopilskoho derzhavnoho tekhnichnoho universytetu, 14*(1), 201–207. [In Ukrainian].

Kharchenko, V. S., Netkacheva, E. I., Orekhova, A. A., et al. (2012). *CASE-otcenka kriticheskikh programmnykh sistem: monografiia*, (In 3 vol.), Vol. 1: Bezopasnost, (Kharchenko, V. S. Scientific Ed.). Kharkov: NAU "KhAI". 301 p. [In Russian].

Cockburn, A. (2001). *Writing Effective Use Cases*. Addison-Wesley. 270 p.

Konorev, B. M. (Ed.), Manzhos, Iu. S., Kharchenko, V. S. (Ed.) et al. (2009). *Invariantno-orientirovannaia otcenka kachestva programmnogo obespecheniia kosmicheskikh sistem: monografiia*. Kharkov: NAU "KhAI". 224 p. [In Russian].

Kornipaev, Ilia. (2013). Trebovaniia dlia programmnogo obespecheniia. Rekomendatcii po sboru i dokumentirovaniiu. Moscow: OZON Status. 118 p. [In Russian].

Kroll, P., & Krachten, F. (2004). Rational Unified Process – eto legko. Rukovodstvo po RUP dlia praktikov : per. s angl. Moscow: KUDITC-OBRAZ. 432 p. [In Russian].

Laplante, Phil. (2009). Requirements Engineering for Software and Systems (1st ed.). Redmond, WA: CRC Press. 268 p.

Lavrishcheva, E. M. (2013). *Software Engineering kompiuternykh sistem. Paradigmy, tekhnologii i CASE-sredstva programmirovaniia*. Kyiv: Naukova dumka. 283 p. [In Russian].

Leffingwell, D., & Widrig, D. (1999). Managing Software Requirements: A Unified Approach. Addison-Wesley Longman Publishing Co., Inc. Boston, MA

Maedche, A., Botzenhardt, A., & Neer, L. (2012). *Software for People: Fundamentals, Trends and Best Practices* (*Management for Professionals*). Springer-Verlag Berlin Heidelberg. 293 p.

Maievskyi, D., & Kozina, Iu. (2015). Gde i kogda formiruetsia kachestvo programmnogo obespecheniia? *Elektrotekhnicheskie i kompiuternye sistemy, 18*, 55–59. [In Russian].

Mansilla, D., Pollo-Cattaneo, M., Britos, P., & García-Martínez, R. (2013). A Proposal of a Process Model for Requirements Elicitation

in Information Mining Projects. *Lecture Notes in Business Information Processing*, *4*, 165–173. doi:10.1007/978-3-642-36611-6_13
Matciashek, L. A. (2002). Analiz trebovanii i proektirovanie sistem. Razrabotka informatcionnykh sistem s ispolzovaniem UML. Moscow: Izd. dom "Viliams". 432 p. [In Russian].
McConnell, S. (2013). *Sovershennyi kod. Master-klass*. Moscow: Izdvo "Russkaia redaktciia". 896 p. [In Russian].
Mishchenko, V. O., Pomorova, O. V., & Hovorushchenko, T. A. (2012). *CASE-otcenka kriticheskikh programmnykh sistem: monografiia*, (In 3 vol.), Vol. 1: Kachestvo. (Kharchenko, V. S. Scientific Ed.). Kharkov: Natc. aerokosmicheskii universitet "KhAI". 201 p. [In Russian].
Myers, G., Badzhett, T., & Sandler, K. (2012). *Iskusstvo testirovaniia programm*, (3rd ed.). Moscow: OOO "ID Viliams". 272 p. [In Russian].
Pomorova, O. V., & Hovorushchenko, T. O. (2013). Intelligent Assessment and Prediction of Software Characteristics at the Design Stage. *American Journal of Software Engineering and Applications* (*AJSEA*), *2*(2), 25–31. Retrieved from: http://article.sciencepublishinggroup.com/pdf/10.11648.j.ajsea.20130202.11.pdf.
Sommerville, I. (2002). *Inzheneriia programmnogo obespecheniia*, (6ix ed.). Moscow: Izd. dom "Viliams". 624 p. [In Russian].
Sutcliffe, A. (1998). Scenario-based requirements analysis. *Requirements Engineering*, *3*(1), 48–65. doi:10.1007/bf02802920
Wiegers, K., & Betti, D. (2014). Development of Software Requirements. (Trans. from English). (3rd Edition). Moscow: Russian Edition; St. Petersburg: BHV-Petersburg. 736 p. [In Russian].
Wiegers, Karl E. (2003). Software Requirements (2nd Edition) (Developer Best Practices). Redmond, WA: Microsoft Press. 544 p.
Yakovyna, V. S. (2012). Vplyv funktsii aktyvatsii RBF neironnoi merezhi na efektyvnist prohnozuvannia kilkosti vidmov prohramnoho zabezpechennia. *Visnyk Natsionalnoho universytetu "Lvivska politekhnika". Seriia: "Kompiuterni nauky ta informatsiini tekhnolohii"*, *732*, 36–39. [In Ukrainian].

Ю. И. Грыцюк, Е. А. Немова

Национальный университет "Львовская политехника", г. Львов, Украина

ОСОБЕННОСТИ ФОРМУЛИРОВАНИЯ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

Предложено языковые особенности процедуры формулирования требований к программному обеспечению (ПО), которые являются общими для любого процесса его разработки. Это позволяет аналитику на основании их возможностей выяснить особенности разработки структуры соответствующего документа и осуществить отбор ключевых требований из допустимого множества. Проанализированы основные возможности требований к ПО и их атрибуты, жестко привязанные к этим требованиям. Результаты такого анализа помогли понять суть предлагаемых принципов формулирования требований к ПО. Выяснены особенности разработки структуры документа с требованиями к ПО и некоторые моменты отбора ключевых требований. Это позволяет его исполнителям доступно излагать в нем нужный материал с завершенностью мысли, осуществлять логическую последовательность представления информации, обеспечивать ее целостность и доступность, ясность для дальнейшего использования и восприятия. Установлено взаимную связь и важность требований к ПО путем их классификации, фильтрации и сортировки. Сделано это для того, чтобы впоследствии получить относительно небольшие наборы требований, каждый из которых касается только одной темы для дальнейшего их анализа. Выявлено языковые особенности процедуры формулирования требований к ПО, уточнены некоторые моменты подготовки соответствующих шаблонов и их детализация. Все это в совокупности ведет к разработке качественного ПО путем тесного сотрудничества бизнес-аналитика с его заказчиком, в результате которой они устанавливают и записывают соответствующие договоренности. Проанализированы критерии, которые нужно использовать для определения качества формулировки требований к ПО, что позволяет аналитику соблюдать некоторые простые правила при их написании.

Ключевые слова: программный проект; требования к программному обеспечению; определение требований; разработка требований; формулирование требований; рецензирование требований; спецификация требований; критерии качества требований; структура документа.

Yu. I. Hrytsiuk, E. A. Nemova

Lviv Polytechnic National University, Lviv, Ukraine

PECULIARITIES OF FORMULATION OF REQUIREMENTS TO THE SOFTWARE

Suggested language features formulating procedures requirements to software, which are common for all of its development process. This allows the analyst on the basis of their ability to clarify the features of the development of the structure of the instrument and to carry out the selection of the key requirements of the feasible set. Analyzed the main features of software requirements and their attributes, are rigidly attached to the bill. The results of this analysis help to understand the essence of the proposed formulation of the principles of software requirements. Has clarified the developing structure of the document with the requirements of the software and some points selection key requirements. This allows its performers available to present it the right material with perfection of thought, carry out a logical sequence of presentation, to ensure its integrity, and availability, clarity for future use and perception. Established mutual relationships and the importance of software requirements by their classification, filtering and sorting. This was done in order to subsequently obtain a relatively small set of requirements, each of which applies only to one subject for further analysis. The proposed approach to determining the importance of software requirements has an additional advantage over other ways, since it allows system analysts to realize their degree of freedom in developing and making the right decision. This means that for individual software requirements, by matching the values of certain system performance indicators, you can get the best values. This approach is very often used when conducting tenders for comparison and evaluation of alternative types of alternatives. Identified language features formulating procedures requirements to the software, to clarify some aspects of the preparation of relevant templates and detailing. All this together leads to the development of quality software through close cooperation business intelligence with its customer, as a result of which they establish and record the relevant agreement. We analyzed the criteria that should be used to determine the quality of the formulation of requirements for software.

Keywords: software project; software requirements; requirements definition; requirements development; requirements formulation; requirements review; requirements specification; requirements quality criteria; document structure.