УДК 004.453
DOI: 10.31891/2219-9365-2019-63-1-45-53

ГОВОРУЩЕНКО Т. О.,
БОДНАР М. А.,
КУШНІР В. О.
Хмельницький національний університет

СУЧАСНІ ПРОБЛЕМИ ФОРМУВАННЯ ТА АНАЛІЗУ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Проведене дослідження впливу інформації у специфікації вимог на якість ПЗ показало, що чинники якості сучасних програмних систем є менш залежними від написання програмного коду, але суттєво залежать від формування та формулювання вимог. Характеристики специфікації вимог до ПЗ значною мірою визначаються характеристиками вхідної інформації (бізнес-вимогами замовника, вимогами предметної галузі, стандартами щодо розроблення ПЗ та стандартами предметної галузі). Аналіз специфікацій вимог до ПЗ показав можливість оцінювання інформації у специфікаціях вимог до ПЗ на предмет її достатності. Дослідження процесу оцінювання достатності інформації у специфікації вимог до ПЗ показало необхідність розроблення частин базової онтології предметної галузі «Інженерія програмного забезпечення» для нефункційних характеристик.

Ключові слова: програмне забезпечення (ПЗ), вимоги до ПЗ, специфікація вимог до ПЗ, достатність інформації у вимогах до ПЗ.

HOVORUSHCHENKO T.,
BODNAR M.,
KUSHNIR V.
Khmelnytskyi National University

MODERN PROBLEMS OF THE FORMATION AND ANALYSIS OF THE SOFTWARE REQUIREMENTS

The conducted study of the impact of information in the software requirements specification (SRS) for software quality has shown that the quality factors of modern software systems are less dependent on the writing of software code, but essentially depend on the formation and formulation of requirements. Characteristics of the SRS are largely determined by the characteristics of the input information (business requirements of the customer, the requirements of the subject domain, standards for software development and industry standards). The analysis of the SRS has shown the possibility to evaluate the information in the SRS on its sufficiency. The study of the process of evaluating the sufficiency of information in the SRS showed the need to develop parts of the base ontology of the subject domain "Software Engineering" for non-functional characteristics.

Keywords: software, software requirements, software requirements specification, sufficiency of information in the software requirements.

Вступ. Практично усі сфери людської діяльності на сьогодні пов’язані з комп’ютерними системами, основою яких є програмне забезпечення (ПЗ). Розроблення ПЗ – це наукомістка діяльність, яка вимагає детального вивчення предметної галузі та повного розуміння цілей продукту, що розробляється.

На сьогодні в світі витрачається більше 250 млрд USD щорічно на розроблення приблизно 175 тис. програмних проектів. Середня вартість проекту для великої компанії становить 2,322 млн USD, для середньої компанії – 1,313 млн USD, а для невеликої компанії – 434 тис. USD [1, 2].

Ключовим фактором забезпечення ефективного застосування ПЗ та однією із основних вимог користувачів і зацікавлених осіб до сучасного ПЗ є досягнення високих значень показників його якості. Якість ПЗ є основним чинником для його успішного впровадження та експлуатації. Потреба у забезпеченні якості ПЗ випливає з того, що помилки та відмови ПЗ загрожують катастрофами, які призводять до людських жертв, екологічних катаклізмів, значних часових втрат та фінансових збитків.

Згідно зі стандартами ISO 25010 [3], ISO 25030 [4], SWEBOK [5], якість ПЗ розглядається як його здатність задовольнити заявлені і передбачувані потреби при його використанні за певних умов. Визначення якості у ISO 9000 [6] та ISO 9001 [7] стосується задоволення вимог, тобто якість у [6, 7] розглядається як характеристика ПЗ, яка відображає ступінь його відповідності вимогам. Визначення якості зі стандартів [6, 7] не враховує факту, що вимоги можуть не відображати повною мірою потреби замовників, тоді задоволення вимог не означатиме задоволення потреб замовників, відтак таке ПЗ не можна буде вважати якісним (насправді матиме місце лише формальне задоволення якості).

Одним з основних чинників, що впливають на якість ПЗ, є якість та достатність інформації нормативної документації (в першу чергу, специфікації вимог до ПЗ).

Отже, успішність реалізації програмного проекту (як вчасне виконання програмного проекту в рамках виділеного бюджету та з реалізацією всіх необхідних можливостей та функцій) суттєво залежить від

ранніх етапів життєвого циклу ПЗ. Враховуючи вищевикладене, *метою даного дослідження* є аналіз сучасних проблем формування та аналізу вимог до ПЗ.

Аналіз впливу інформації у специфікації вимог на якість програмного забезпечення. У галузі забезпечення якості ПЗ і досі існують проблеми, які були помітними ще більше 50 років тому – великі проекти виконуються з відставанням від графіка або з перевищенням кошторису витрат, розроблений продукт не має необхідних функціональних можливостей, продуктивність його часто є низькою, якість ПЗ не влаштовує споживачів [8]. Аналітичні дослідження та огляди щодо ПЗ, які виконували протягом кількох останніх років провідні зарубіжні аналітики, підтверджують ці, не дуже обнадійливі, результати [9]. Так, за наявності ряду методів та засобів, залученні кращих фахівців для розроблення технологій та стандартів забезпечення якості програмних комплексів, якість ПЗ і досі залежить від знань та досвіду розробників [8, 9].

Статистика успішності програмних проектів за 1994–2015 роки, за даними The Standish Group International (Chaos reports) [1, 2, 10], представлена на рис. 1. Успішними вважаються проекти, які були виконані вчасно, в рамках бюджету, з необхідними можливостями та функціями. Проблемними вважаються проекти, які мали перевищення термінів, перевитрати або не мали необхідних можливостей та функцій. Провальними вважаються проекти, які були скасовані до завершення, або були доставлені, але ніколи не використовуються.

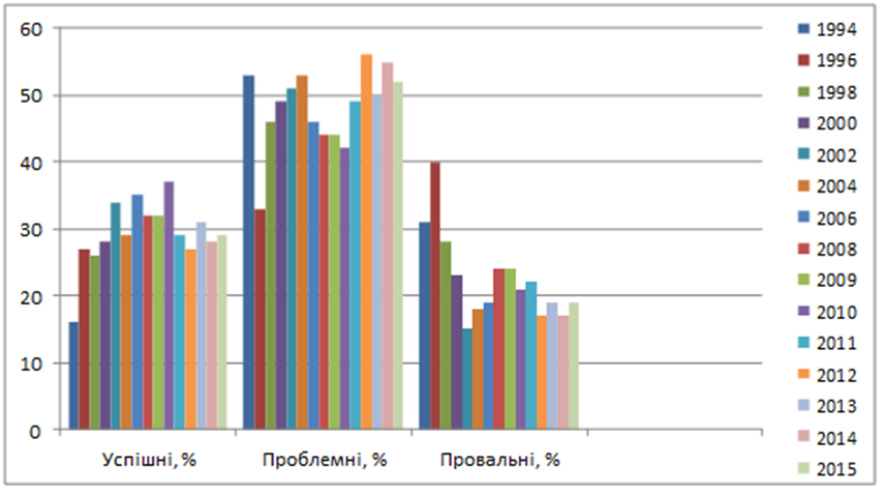

Рис. 1. Статистика успішності програмних проектів у 1994–2015 рр.

Аналіз даних рис. 1 дав можливість побачити спад кількості успішних проектів та приріст кількості провальних проектів у 2014-2015 роках (порівняно з 2013 р.), але в той же час кількість проблемних проектів є досить сталою величиною у 2012–2015 роках та складає 50% і більше проектів. Крім цього, на рис.1 можна побачити своєрідну періодичність (синусоїдальність) приростів та спадів кількостей для всіх груп проектів, тому не виключено, що в наступні роки може відбутись спад кількості успішних проектів і приріст кількості провальних проектів.

Дослідження The Standish Group International доводить, що 31,1% програмних проектів будуть скасовані до їх завершення, а 52,7% проектів коштуватимуть 189% від їхніх початкових оцінок. Але до перевитрат при розробленні додаватимуться ще альтернативні витрати при використанні такого ПЗ, які можуть сягати трильйонів доларів. Так, наприклад, відсутність надійного ПЗ для обробки багажу в аеропорту м. Денвер коштує місту 1,1 млн доларів на день [1, 2].

В середньому, лише 16–29% програмних проектів виконуються в межах запланованих часу та бюджету (для великих компаній – 9–16% проектів). Крім цього, проекти, виконані найбільшими американськими компаніями, мають лише приблизно 42% від необхідних можливостей та функцій (для малих компаній 78,4% програмних проектів мають принаймні 74,2% початкових можливостей та функцій) [1, 2] – рис. 2 [2].

Серед основних причин можливих невдач називають [11]: 1) нечітке й неповне формування та формулювання вимог до ПЗ; 2) часту зміну вимог і специфікацій; 3) некоректне розуміння або недостатній аналіз проекту.

Помилки формування та формулювання вимог і проектування архітектури складають 25–55% всіх помилок, причому чим більший обсяг ПЗ, тим більше помилок вноситься саме на ранніх етапах [11].

Аналіз великої кількості програмних проектів, проведений у [12], підтвердив той факт, що головне місце виникнення помилок у ПЗ – це етап формування та формулювання вимог (специфікація), друге місце по внесенню помилок «посідає» етап проектування архітектури. У більшості випадків помилки вказують на проблеми специфікації та проектування архітектури, тобто фактично вже в кінці етапу проектування архітектури можна виявити та усунути левову частку всіх помилок ПЗ.

Основні причини, які роблять етап формування та формулювання вимог основним «постачальником» помилок: 1) відсутність специфікації; 2) неповнота або суперечливість інформації специфікації; 3) часті зміни специфікації.

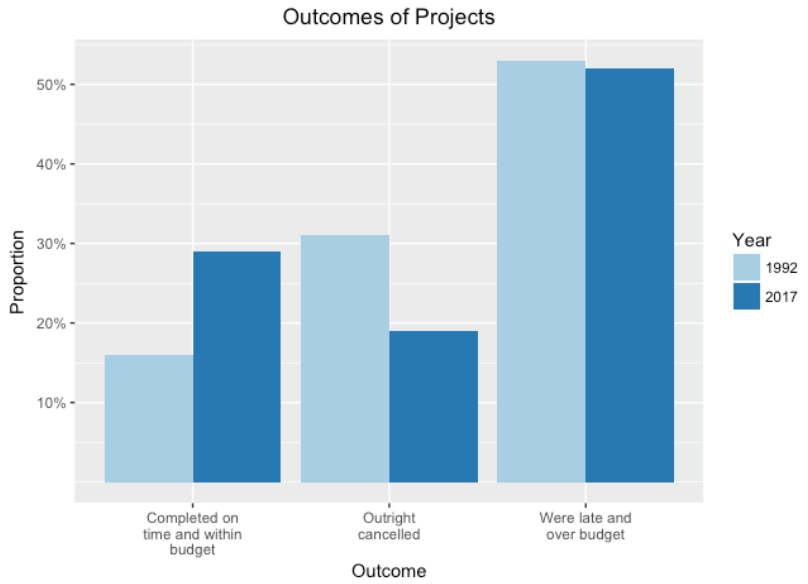

Рис. 2. Порівняльна статистика успішності про-грамних проектів у 1992 та 2017 роках, за даними The Standish Group International [2]

В роботах [13–15] підтверджується факт, що причини майже всіх інцидентів та катастроф, пов'язаних з ПЗ, криються у специфікації вимог. Переважна більшість аварій, пов'язаних із ПЗ, виникли через помилкові вимоги, а не через помилки кодування. У [15] описано результати експерименту, проведеного для підтвердження або відкидання гіпотези про те, що збої та помилки ПЗ, написаного різними розробниками за однією специфікацією, статистично незалежні. Під час експерименту декілька незалежних груп розробників писали свою версію ПЗ за однією специфікацією. В результаті такого експерименту було встановлено, що версії ПЗ, написані різними розробниками за однаковими вимогами, містили ряд спільних помилок, пов'язаних із помилками або неточностями вимог (специфікації).

Дефекти вимог бажано виявляти та усувати до того, як вони почнуть впливати на більш пізні етапи розроблення. Ранні етапи життєвого циклу впливають на якість ПЗ більше, ніж пізні, тому час, витрачений на контроль якості на ранніх етапах, забезпечує можливість знизити рівень дефектів, скоротити терміни розроблення та знизити витрати на більш пізніх етапах.

Чим раніше буде виявлено дефект (помилка, порушення, недолік, несправність), тим дешевше обійдеться його виправлення. Як показано на рис. 3, витрати на виправлення дефектів, виявлених після випуску продукту, майже в 100 разів перевищують витрати на виправлення, якщо недоліки були виявлені в процесі формування та формулювання вимог [16]. Якщо розглянути альтернативні варіанти дизайну та провести аналіз впливу ще до створення системи як такої, це дасть суттєву економію коштів на корекцію помилок (рис. 4).

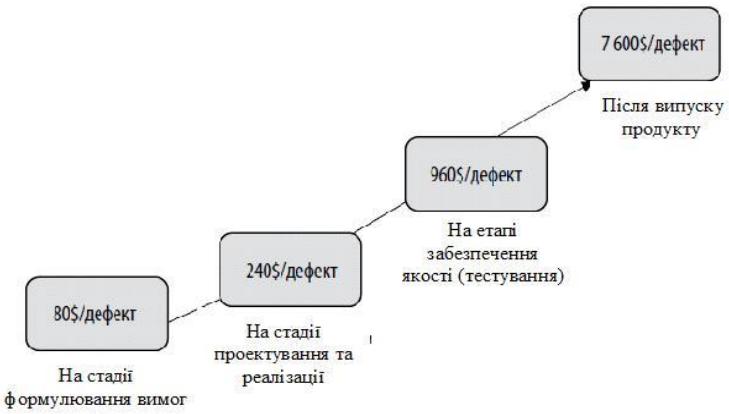

Рис. 3. Зростання вартості виправлення дефектів в процесі розроблення програмного забезпечення [16]

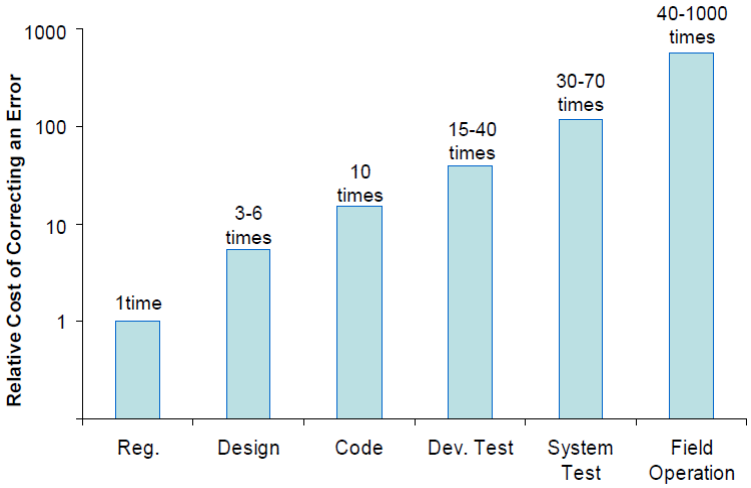

Рис. 4. Витрати часу на корекцію помилок протягом життєвого циклу ПЗ [16]

Із збільшенням інтервалу між моментами внесення та виявлення дефекту вартість його виправлення сильно зростає. Чим довше помилка зберігається у ланцюгу розроблення ПЗ, тим сильніше вона проникає в інші частини ПЗ, тим більше шкоди завдає на наступних етапах, і тим більше коштів доведеться витратити на її усунення.

В процесі формування та формулювання вимог можуть відбуватись інформаційні втрати через неповне та різне розуміння потреб та контексту інформації – особливо такі втрати суттєві для програмних проектів, які розробляються на стику предметних галузей (наприклад, ПЗ для медицини), коли враховувати потрібно як стандарти щодо розроблення ПЗ, так і стандарти предметної галузі, для якої розробляється ПЗ. Таку кількість стандартів важко імплементувати, а ще важче оцінити ступінь врахування рекомендацій цих стандартів.

З визначення якості ПЗ як ступеня задоволеності користувачів або ступеня відповідності ПЗ потребам замовників слідує, що, якщо цілі проекту, встановлені на ранніх етапах життєвого циклу, не відповідають потребам користувачів, то ПЗ неможливо визнати якісним, навіть якщо при його розробленні були використані сучасні технології та були задіяні найкваліфікованіші розробники. Отже, якість та успішність реалізації програмного проекту суттєво залежать від специфікації вимог, а також від наявної у ній інформації. Таке експериментальне свідчення безпосередньо призводить до необхідності поглиблення аналізу специфікації.

Аналіз інформації у специфікації вимог до ПЗ забезпечує можливість вибору програмного проекту з позицій його прогнозованої якості, підвищує ефективність управління проектом за рахунок обґрунтованості рішень, скорочує час на їх розробку і прийняття, зменшує витрати на збір і обробку відомостей. Недостатність або суперечливість інформації у специфікації вимог до ПЗ знижує результативність та достовірність розроблення та оцінювання ПЗ.

Програмні вимоги визначають потрібні характеристики ПЗ, а також впливають на методи кількісного оцінювання та сформульовані для оцінювання цих характеристик критерії приймання. Отже, всю необхідну інформацію закладено вже у специфікації вимог до ПЗ, тобто вже на основі специфікації вимог до ПЗ можна оцінити достатність інформації для подальшого розроблення ПЗ. Якщо деякі інформаційні елементи, зазначені у [17], відсутні, то у специфікації вимог недостатньо інформації для подальшого розроблення ПЗ, і розробники повинні внести необхідні доповнення у специфікацію.

Дослідження відомих методів аналізу специфікацій вимог до ПЗ показало, що наразі розроблено методи аналізу специфікації вимог (Using natural language processing technique, Using CASE analysis method, QAW-method, Using global analysis method, O'Brien's approach, Method to discover missing requirement elicitation, Selection of requirements elicitation technique, Comparison and categorization of requirements elicitation techniques, Techniques for ranking and prioritization of software requirements) [12, 18, 19], але перераховані методи спрямовані на контроль за реалізацією вимог, а не на оцінювання достатності інформації у специфікації вимог до ПЗ.

Аналіз відомих засобів показав, що розроблено ряд засобів, зокрема, автоматизовані засоби аналізу специфікації вимог (IBM Rational RequisitePro, IBM Rational/Telelogic DOORS, Borland Caliber RM, Sybase PowerDesigner, Open Source Requirements Management Tool, Sigma Software, DEVPROM) [12, 18, 19], але ці засоби не орієнтовані на оцінювання достатності інформації у специфікації вимог до ПЗ.

Отже, як показав проведений аналіз впливу інформації специфікації вимог на якість ПЗ, чинники якості сучасних програмних систем є менш залежними від написання програмного коду, але суттєво залежать від

формування та формулювання вимог і проектування архітектури. Дефекти, внесені на етапах формування та формулювання вимог і проектування архітектури, варто виявляти та усувати до того, як вони почнуть впливати на результати більш пізніх етапів життєвого циклу. Якість та успішність реалізації програмного проекту суттєво залежать від специфікації вимог до ПЗ.

За таких умов *актуальним і дуже важливим* є аналіз специфікації вимог до ПЗ, можливість «відсікти» програмні проекти з неповною (з недостатньою інформацією) специфікацією. Але відомі методи та засоби не розв'язують проблему оцінювання достатності інформації у специфікації вимог до ПЗ.

Формування специфікації вимог до програмного забезпечення. Життєвий цикл розроблення ПЗ починається з формування комплексу вимог та специфікації вимог до ПЗ на їх основі. Основними джерелами інформації при формуванні вимог до ПЗ є бізнес-вимоги замовника, вимоги предметної галузі, стандарти, описи процесу розроблення та впровадження подібного ПЗ і т.і.

Бізнес-вимоги описують мету створення системи, критерії досягнення цієї мети, ключові вимоги до результатів та їх пріоритети і обмеження.

Розглянемо детальніше вхідну інформацію при формуванні вимог до ПЗ на прикладі вимог до облікової Enterprise-системи [20]. Вся документація на цей продукт складалась з загальної частини, вимог, опису реалізації, тестування, посібників та управління.

Загальна частина документації складалась з двох розділів: списку термінів і їх визначень та опису бізнес-ролей користувачів. Будь-яка документація щодо системи, включаючи, наприклад, тестові сценарії, спиралася на наведені в цьому розділі визначення. В списку термінів до облікової Enterprise-системи виявилось близько 200 бізнес- і системних визначень. Список бізнес-ролей використовується для реалізації груп і ролей користувачів, призначення функціональних прав, він необхідний тестувальникам, щоб тестувати сценарій під потрібними ролями. Опис ролей було дано на якісному рівні в текстовій формі на основі аналізу основних функцій працівників [20].

Розділ вимог документації до облікової Enterprise-системи включав бізнес-вимоги (загальні сценарії, сценарії використання, алгоритми і перевірки), системні вимоги, нефункційні вимоги, вимоги до інтеграції, вимоги до інтерфейсу. Бізнес-вимоги описували те, що необхідно бізнес-користувачам. Наприклад, їм зовсім не потрібний об'єкт системи «Користувач», але зате їм потрібно мати можливість змінити вартість товару в рахунку та роздрукувати його. Бізнес-вимоги складались з загальних сценаріїв, сценаріїв використання (use cases) і опису алгоритмів обробки даних. Вимоги були представлені у вигляді дерева (з циклами). Тобто загальні сценарії уточнювалися сценаріями використання, які, в свою чергу, мали посилання на перевірки і алгоритми [20].

Коренева сторінка дерева вимог до облікової Enterprise-системи складалась з загальних сценаріїв, кожен з яких описував один з 24 бізнес-процесів, що підлягали реалізації. Загальний сценарій – це послідовність кроків користувача і системи для досягнення певної мети, основна мета його – узагальнити сценарії використання і побачити, що хоче зробити користувач, і як система йому в цьому допомагає. Загальні сценарії також містили кроки, які користувач здійснював поза системою, оскільки треба було відобразити його роботу у всій повноті, з усіма етапами, необхідними для досягнення бізнес-цілей [20].

Сценарій використання містив пронумеровані кроки. Кожен крок — це, зазвичай, просте речення в теперішньому часі [20].

При написанні алгоритмів аналітик намагався їх описати якомога більш повно. Проте отримані тексти виявлялися погано читабельними, і, як правило, все одно якісь деталі втрачалися. Тому аналітику варто описувати алгоритм настільки повно, наскільки це важливо в плані бізнес-логіки, другорядні перевірки програміст сам зобов'язаний передбачити в коді [20].

Загальна частина та частина вимог документації на продукт складає вхідну інформацію при формуванні вимог до ПЗ. Уся ця інформація, як видно з вищенаведеного прикладу, представляється у вигляді словесного опису бізнес-процесів на рівнях бізнес-функцій та операцій для різних функційних частин ПЗ, наборів бізнес-правил та набору моделей предметної галузі, наприклад, концептуальних моделей, сценаріїв, діаграм та ін.

Крім цього, при формуванні вимог до ПЗ необхідно враховувати представлені у текстовій формі вимоги стандартів щодо розроблення ПЗ та стандартів предметної галузі, для якої розробляється ПЗ (рис. 5).

На основі вищевказаної вхідної інформації формується специфікація вимог до ПЗ.

Отже, характеристики специфікації вимог до ПЗ значною мірою визначаються характеристиками вхідної інформації. Якщо вхідна інформація є недостатньою, або ж вона є неточною, неоднозначною чи суперечливою, то існує висока імовірність, що усі ці недоліки проявляться і у специфікації вимог до ПЗ. А програмні проекти, специфікація вимог яких містять недостатню, неточну, неповну, суперечливу інформацію, не можуть мати успішної реалізації, як було доведено вище.

Для забезпечення якості ПЗ необхідно здійснити дослідження характеристик вхідної інформації з метою виявлення та усунення проблем і недоліків на початкових етапах життєвого циклу ПЗ. В процесі такого дослідження необхідно оцінити, наскільки повно у вхідній інформації, зокрема, у бізнес-вимогах,

відображена інформація, що описує функції, властивості та обмеження майбутнього ПЗ, особливо ті, що характеризують його нефункційні характеристики, такі як якість, надійність, гарантоздатність та ін., і виявити факти недостатності інформації, котра має до них відношення.

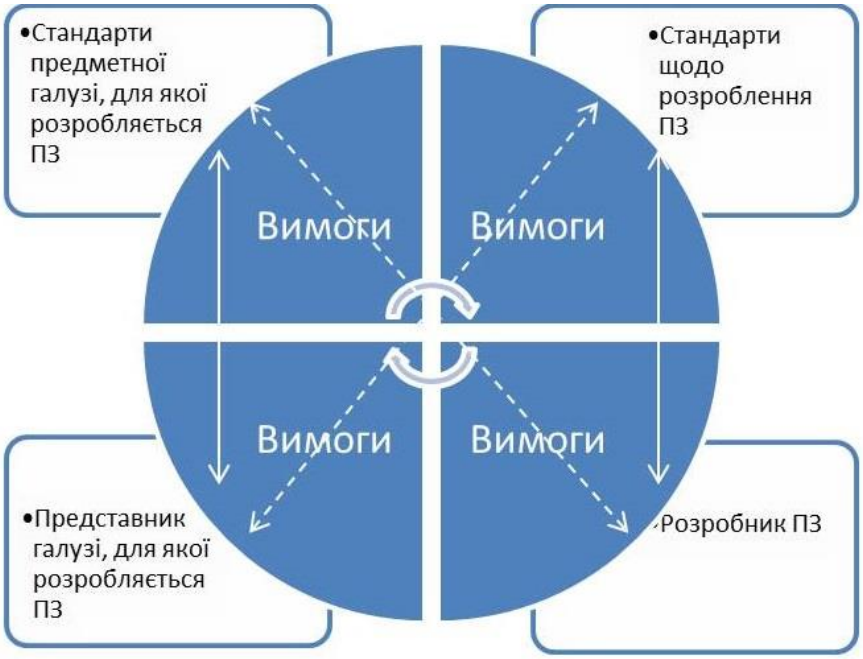

Рис. 5. Врахування вимог стандартів щодо розроблення ПЗ та стандартів предметної галузі, для якої розробляється ПЗ

Наприклад, інформаційні потоки, що описують вимоги, які регламентують характеристики якості, формуються на основі високорівневих цілей організації замовників та цілей і задач, а також на основі бізнес-правил, які включають корпоративну політику та вимоги промислових стандартів, представлених у вигляді документу про об'єкт і межі проекту, обмеження дизайну і реалізації. Такі інформаційні потоки складаються з вимог, які визначають характеристики якості – наприклад, вимога «Як мінімум 25% пропускної спроможності процесора і оперативної пам'яті, доступної застосуванню, не повинні використовуватися в умовах запланованого пікового навантаження» регламентує таку характеристику якості, як ефективність і має наступне представлення у специфікації вимог до ПЗ: «Середнє значення пропускної здатності – 75%», «Максимальне використання пам'яті – 75%», «Ліміт завантаження – пікове навантаження»; вимога «Тільки користувачі, що володіють привілеями рівня Аудитор, повинні мати можливість переглядати транзакції клієнтів» регламентує таку характеристику якості, як захищеність і має наступне представлення у специфікації вимог до ПЗ: «Керованість доступністю – рівні привілеїв», а також на основі вимог такого типу підраховуються такі вимоги специфікації, як «Кількість подій, що вимагають певного типу власності», «Кількість наданих методів аутентифікації».

Аналіз специфікації вимог до програмного забезпечення. Сьогодні оцінювання атрибутів для нефункційних характеристик ПЗ відбувається лише на етапі оцінювання ПЗ для готового програмного коду [21]. Але всі необхідні атрибути та показники закладено вже у специфікації вимог до ПЗ [21], тобто на основі специфікації вимог до ПЗ можна оцінити достатність інформації щодо майбутнього забезпечення нефункційних характеристик ПЗ. Якщо деякі атрибути відсутні, то у специфікації вимог недостатньо інформації для тієї чи іншої нефункційної характеристики. Для усунення недостатності інформації до розробників вхідної інформації необхідно сформувати повторний запит щодо вимог, які регламентують таку нефункційну характеристику, на основі якого вони повинні внести необхідні доповнення у вхідну інформацію при формуванні вимог до ПЗ.

Одним з підходів виявлення факту недостатності інформації у специфікації вимог до ПЗ є оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ на основі порівняльного аналізу онтологій [9]. В рамках такого підходу розроблено теоретичні та прикладні засади інформаційної технології оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ, зокрема, розроблено систему оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ на основі порівняльного аналізу онтологій.

Дана система працює наступним чином: 1) генерує та наповнює шаблони онтологій для визначення якості конкретного ПЗ (за стандартом ISO 25010 [3]); 2) порівнює онтології для визначення якості конкретного ПЗ з відповідними базовими онтологіями предметної галузі «Інженерія програмного

забезпечення» (частина «Якість ПЗ»); 3) на основі порівняння онтологій, враховуючи критерій достатності інформації щодо якості у специфікаціях вимог до ПЗ, робить висновок про достатність або недостатність інформації щодо якості у специфікації вимог до конкретного ПЗ; 4) якщо інформації щодо якості у специфікації достатньо, то продовжується робота над проектом за цією специфікацією; 5) якщо інформації щодо якості у специфікації недостатньо, то формується запит на додавання інформації щодо вимог, які регламентують характеристики якості, у вхідну інформацію при формуванні вимог до ПЗ. Цей запит містить перелік атрибутів, для зазначення яких у специфікації вимог не вистачає інформації у бізнес-вимогах, а також рекомендовану пріоритетність доповнення цих атрибутів у специфікацію (в залежності від важливості та вагомості того чи іншого атрибута). На наступному кроці здійснюється ітерація доповнення специфікації інформацією щодо якості ПЗ.

Схема процесу оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ представлена на рис. 6.

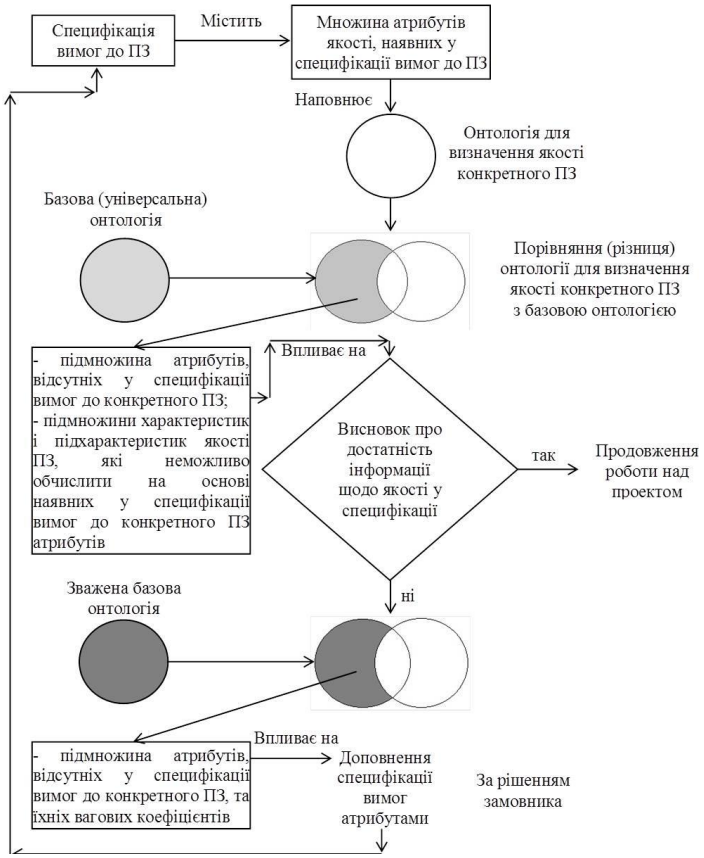

Рис. 6. Схема процесу оцінювання достатності інформації щодо якості у специфікаціях вимог до ПЗ

За аналогією, на основі порівняльного аналізу онтологій, можна оцінювати достатність інформації щодо решти нефункційних характеристик у специфікаціях вимог до ПЗ, але для цього потрібно розробити частини базової онтології предметної галузі «Інженерія програмного забезпечення» для нефункційних характеристик, використовуючи відповідні стандарти, що регламентують атрибути, на основі яких відбувається визначення та оцінювання тієї чи іншої нефункційної характеристики, на що й будуть спрямовані подальші зусилля авторів.

Висновки. Проведене дослідження впливу інформації у специфікації вимог на якість ПЗ показало, що чинники якості сучасних програмних систем є менш залежними від написання програмного коду, але суттєво залежать від формування та формулювання вимог і проектування архітектури. Ризиками недостатньо відпрацьованого етапу формування та формулювання вимог є недотримання термінів проектів та фінансові перевитрати, що можуть призвести до закриття проекту, а то й розпаду софтверної компанії внаслідок її фінансової нестабільності.

Як доведено вище, саме в кінці етапу проектування архітектури можна й варто виявляти та усувати до 55% всіх помилок майбутнього ПЗ. Дефекти, внесені на етапах формування та формулювання вимог і проектування архітектури, варто виявляти та усувати до того, як вони почнуть впливати на результати більш пізніх етапів життєвого циклу. Якість та успішність реалізації програмного проекту суттєво залежать від специфікації вимог до ПЗ.

Бізнес-вимоги замовника, вимоги предметної галузі, стандарти щодо розроблення ПЗ та стандарти предметної галузі складають вхідну інформацію при формуванні вимог до ПЗ, що описує функції майбутнього ПЗ, а також його властивості та обмеження. На основі такої вхідної інформації формується специфікація вимог до ПЗ. Отже, характеристики специфікації вимог до ПЗ значною мірою визначаються характеристиками вхідної інформації. Тому, для забезпечення якості ПЗ необхідно здійснити дослідження характеристик вхідної інформації з метою виявлення та усунення проблем і недоліків на початкових етапах життєвого циклу ПЗ. В процесі такого дослідження необхідно оцінити, наскільки повно у вхідній інформації відображена інформація щодо нефункційних вимог, і виявити факти недостатності інформації, котра має до них відношення.

Аналіз специфікацій вимог до ПЗ показав можливість оцінювання інформації у специфікаціях вимог до ПЗ на предмет її достатності. В разі встановлення факту недостатності інформації у специфікаціях вимог до ПЗ, до розробників вхідної інформації надходить повторний запит на додавання інформації щодо вимог, які регламентують нефункційні характеристики. Запит містить перелік атрибутів, для зазначення яких у специфікації вимог не вистачає інформації у бізнес-вимогах, а також рекомендовану пріоритетність доповнення цих атрибутів у специфікацію (залежно від важливості та вагомості того чи іншого атрибута). На наступному кроці здійснюється ітерація доповнення специфікації інформацією (атрибутами).

Дослідження процесу оцінювання достатності інформації у специфікації вимог до ПЗ показало необхідність розроблення частин базової онтології предметної галузі «Інженерія програмного забезпечення» для нефункційних характеристик на основі відповідних стандартів, на що й будуть спрямовані подальші зусилля авторів.

Література

1. Hastie Shane. Standish Group 2015 Chaos Report – Q&A with Jennifer Lynch [Electronic resource] / Shane Hastie, Stéphane Wojewoda. – Access mode: http://www.infoq.com/articles/standish-chaos-2015 (viewed on October 25, 2018).
2. A Look at 25 Years of Software Projects. What Can We Learn? [Electronic resource]. – Access mode: https://speedandfunction.com/look-25-years-software-projects-can-learn/ (viewed on October 25, 2018).
3. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models: ISO/IEC 25010:2011. – [Introduced 01.03.2011]. – Geneva (Switzerland): ISO, 2011. – 34 p. – (International standard).
4. Software engineering. Software product Quality Requirements and Evaluation (SQuaRE). Quality requirements: ISO/IEC 25030:2007. – [Introduced 01.06.2007]. – Geneva (Switzerland): ISO, 2007. – 36 p. – (International standard).
5. Software Engineering. Guide to the software engineering body of knowledge (SWEBOK): ISO/IEC TR 19759:2015. – [Introduced 01.10.2015]. – Geneva (Switzerland): ISO, 2015. – 336 p. – (International standard).
6. Quality management systems. Fundamentals and vocabulary: ISO 9000:2015. – [Introduced 15.09.2015]. – Geneva (Switzerland): ISO, 2015. – 51 p. – (International standard).
7. Quality management systems. Requirements: ISO 9001:2015. – [Introduced 15.09.2015]. – Geneva (Switzerland): ISO, 2015. – 29 p. – (International standard).
8. Мищенко В. О. CASE-оценка критических программных систем: у 3 т. Т. 1 : Качество : монография / В. О. Мищенко, О.В. Поморова, Т. А. Говорущенко. Под ред. В. С. Харченко. – Харьков: Нац. аэрокосмический университет «ХАИ», 2012. – 201 с.
9. Говорущенко Т. О. Методологія оцінювання достатності інформації для визначення якості програмного забезпечення : монографія / Т. О. Говорущенко. – Хмельницький : Хмельницький національний університет, 2017. – 310 с.
10. The Standish Group International: CHAOS Manifesto – Think big, act small. Technical report, CHAOS Knowledge Center (2013) [Electronic resource] – Access mode: http://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf (viewed on October 25, 2018).
11. McConnell, S. Code complete / S. McConnell. – Microsoft Press, 2013. – 896 p.
12. Jones C. The economics of software quality / C.Jones, O.Bonsignour. – Boston: Pearson Education, 2012. – 588 p.
13. Levenson N. G. Engineering a safer world: systems thinking applied to safety / N. G. Levenson. – MIT Press, 2012. – 560 p.
14. Ishimatsu T. Hazard analysis of complex spacecraft using systems-theoretic process analysis / T. Ishimatsu, N. G. Levenson, J. P. Thomas, C.H. Fleming, M. Katahira, Yu. Miyamoto, R. Ujiie, H. Nakao, N. Hoshino // Journal of Spacecraft and Rockets. – 2014. – Vol. 51. – No. 2. – P. 509-522.
15. Levenson N. Software challenges in achieving space safety / N. Levenson // Journal of the British Interplanetary Society. – 2009. – Vol. 62. – P. 265-272.
16. Shamieh C. Systems Engineering for Dummies / C. Shamieh. – New York: Wiley Publishing, 2011. – 68 p.
17. Systems and software engineering. Life cycle processes. Requirements engineering: ISO/IEC/IEEE 29148:2011. – [Introduced 01.12.2011]. – Geneva (Switzerland): ISO, 2011. – 28 p. – (International standard).
18. Sommerville I. Software Engineering / I. Sommerville. – London : Pearson, 2015. – 816 p.
19. Jones C. Software assessments, benchmarks, and best practices / C. Jones. – Addison-Wesley, 2000. – 688 p.
20. Стасевич А. Приклад написання функціональних вимог до Enterprise-системи [Електронний ресурс] / А. Стасевич. – Режим доступу : http://it-ua.info/news/2014/12/11/priklad-napisannya-funkconalnih-vimog-do-enterprise-sistem.html (дата звертання 25.10.2018).
21. Маевский Д. Где и когда формируется качество программного обеспечения? / Д. Маевский, Ю. Козина // Электротехнические и компьютерные системы. – 2015. – № 18. – С. 55–59.

References

1. Hastie Shane. Standish Group 2015 Chaos Report – Q&A with Jennifer Lynch [Electronic resource] / Shane Hastie, Stephane Wojewoda. – Access mode: http://www.infoq.com/articles/standish-chaos-2015 (viewed on October 25, 2018).
2. A Look at 25 Years of Software Projects. What Can We Learn? [Electronic resource]. – Access mode: https://speedandfunction.com/look-25-years-software-projects-can-learn/ (viewed on October 25, 2018).
3. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). System and software quality models: ISO/IEC 25010:2011. – [Introduced 01.03.2011]. – Geneva (Switzerland): ISO, 2011. – 34 p. – (International standard).
4. Software engineering. Software product Quality Requirements and Evaluation (SQuaRE). Quality requirements: ISO/IEC 25030:2007. – [Introduced 01.06.2007]. – Geneva (Switzerland): ISO, 2007. – 36 p. – (International standard).
5. Software Engineering. Guide to the software engineering body of knowledge (SWEBOK): ISO/IEC TR 19759:2015. – [Introduced

01.10.2015]. – Geneva (Switzerland): ISO, 2015. – 336 p. – (International standard).

6. Quality management systems. Fundamentals and vocabulary: ISO 9000:2015. – [Introduced 15.09.2015]. – Geneva (Switzerland): ISO, 2015. – 51 p. – (International standard).

7. Quality management systems. Requirements: ISO 9001:2015. – [Introduced 15.09.2015]. – Geneva (Switzerland): ISO, 2015. – 29 p. – (International standard).

8. Mishenko V. O. CASE-ocenka kriticheskih programmnyh sistem: u 3 t. T. 1 : Kachestvo : monografiya / V. O. Mishenko, O.V. Pomorova, T. A. Govorushenko. Pod red. V. S. Harchenko. – Harkov: Nac. aerokosmicheskij universitet «HAI», 2012. – 201 s.

9. Govorushenko T. O. Metodologiya ocinyuvannya dostatnosti informaciyi dlya viznachennya yakosti programnogo zabezpechennya: monografiya / T. O. Govorushenko. – Khmelnytskyi : Khmelnytskyi National University 2017. – 310 s.

10. The Standish Group International: CHAOS Manifesto – Think big, act small. Technical report, CHAOS Knowledge Center (2013) [Electronic resource]. – Access mode: http://www.versionone.com/assets/img/files/CHAOSManifesto2013.pdf (viewed on October 25, 2018).

11. McConnell, S. Code complete / S. McConnell. – Microsoft Press, 2013. – 896 p.

12. Jones C. The economics of software quality / C.Jones, O.Bonsignour. – Boston: Pearson Education, 2012. – 588 p.

13. Levenson N. G. Engineering a safer world: systems thinking applied to safety / N. G. Levenson. – MIT Press, 2012. – 560 p.

14. Ishimatsu T. Hazard analysis of complex spacecraft using systems-theoretic process analysis / T. Ishimatsu, N. G. Levenson, J.P. Thomas, C. H. Fleming, M. Katahira, Yu. Miyamoto, R. Ujiie, H. Nakao, N. Hoshino // Journal of Spacecraft and Rockets. – 2014. – Vol. 51. – No. 2. – Pp. 509-522.

15. Levenson N. Software challenges in achieving space safety / N. Levenson // Journal of the British Interplanetary Society. – 2009. – Vol. 62. – P. 265-272.

16. Shamieh C. Systems Engineering for Dummies / C. Shamieh. – New York: Wiley Publishing, 2011. – 68 p.

17. Systems and software engineering. Life cycle processes. Requirements engineering: ISO/IEC/IEEE 29148:2011. – [Introduced 01.12.2011]. – Geneva (Switzerland): ISO, 2011. – 28 p. – (International standard).

18. Sommerville I. Software Engineering / I. Sommerville. – London: Pearson, 2015. – 816 p.

19. Jones C. Software assessments, benchmarks, and best practices / C. Jones. – Addison-Wesley, 2000. – 688 p.

20. Stasevich A. Priklad napisannya funkcionalnih vimog do Enterprise-sistemi [Elektronnij resurs] / A. Stasevich. – Rezhim dostupu: http://it-ua.info/news/2014/12/11/priklad-napisannya-funkconalnih-vimog-do-enterprise-sistem.html (data zvertannya 25.10.2018).

21. Maevskij D. Gde i kogda formiruetsya kachestvo programmnogo obespecheniya? / D. Maevskij, Yu. Kozina // Elektrotehnicheskie i kompyuternye sistemy. – 2015. – № 18. – S. 55-59.

Рецензія/Peer review : 13.10.2018 Надрукована/Printed : 05.02.2019
Рецензент: д. т. н., проф. Бармак О. В.