4. ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Науковий вісник НЛТУ України
Scientific Bulletin of UNFU
http://nv.nltu.edu.ua
https://doi.org/10.15421/40270433
Article received 13.05.2017 p.
Article accepted 24.05.2017 p.
УДК 004.414.3:681.322:855.5[075.8]

ISSN 1994-7836 (print)
ISSN 2519-2477 (online)

@ Correspondence author
Yu. I. Grytsiuk
yurii.i.hrytsiuk@lpnu.ua

Ю. І. Грицюк, І. Ф. Лешкевич

Національний університет "Львівська політехніка", м. Львів, Україна

ОСОБЛИВОСТІ ВИЗНАЧЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПРОБЛЕМИ ЇХ АНАЛІЗУ

Проведено системний аналіз предметної області, результати якої дали змогу достовірно описати нагальні проблеми, які стосуються процедур визначення та аналізу вимог до програмного забезпечення (ПЗ), з'ясовано види діяльності аналітика для ефективного вирішення цих проблем, а також запропоновано основні показники якості майбутнього ПЗ, удосконалено формалізовану модель для реалізації процедур визначення та аналізу вимог до ПЗ. З'ясовано, що успішне функціонування ПЗ багато в чому залежить від правильної організації процесу виконання робіт з визначення та аналізу вимог до нього. При цьому необхідні всі відповідні компетенції потенційні фахівці-аналітики отримують у процесі вирішення конкретних завдань у різних предметних областях як під час навчання, так і протягом виробничої діяльності. Встановлено, що процедура аналізу вимог до ПЗ може бути тривалою та ресурсовитратною, що вимагає від аналітиків детального знання предметної області та застосування тонких психологічних навиків. З'ясовано, що багатьма науковцями і практиками виокремлено такі основні етапи процедури аналізу вимог до ПЗ: визначення зацікавлених сторін у розробленні ПЗ; інтерв'ю із зацікавленими сторонами щодо їхніх потреб від майбутнього ПЗ; спільні сесії визначення вимог до ПЗ; набори вимог до ПЗ у стилі контракту; вимірювані цілі застосування майбутнього ПЗ; прототипи – макети майбутнього ПЗ; прецеденти – технологія документування вимог до ПЗ; специфікація вимог до ПЗ.

Ключові слова: предметна область; системний аналіз предметної області; види діяльності аналітика; визначення вимоги до програмного забезпечення (ПЗ); аналіз вимоги до ПЗ; показники якості ПЗ; модель визначення та аналізу вимог до ПЗ.

Вступ. Успішне функціонування програмного забезпечення (ПЗ) багато в чому залежить від правильної організації процесу виконання робіт з визначення та аналізу вимог до нього (Vigers, 2004). Члени команди розробників ПЗ можуть самостійно визначати вимоги або залучати фахівця-аналітика. У будь-якому випадку зазначена діяльність вимагає не тільки знання інженерії ПЗ (Lavrishheva & Petruhin, 2006), а й певного життєвого та професійного досвіду, вміння спілкуватися із представниками від зацікавлених сторін і, що найважливіше, аналітичних здібностей виконавців такої роботи (Leffingujell & Uidrig, 2002). Усі ці компетенції потенційні фахівці-аналітики отримують у процесі вирішення конкретних завдань у різних предметних областях як під час навчання, так і протягом виробничої діяльності (Kobern, 2011).

Для засвоєння особливостей виконання робіт з визначення та аналізу вимог до ПЗ у навчальних закладах передбачені як лекційні, так і практичні заняття (Vigers, 2004). Проте, проблема полягає в тому, що вони дають змогу студенту оволодіти необхідним набором теоретичних знань і деяким практичним вмінням, але не дають практичних навиків, тобто можливостей змоделювати його взаємодію із представниками організації-замовника ПЗ. Пов'язано це з тим, що проведення практичних занять у реальному середовищі організації-замовника трапляється вкрай рідко через часту відмову самих організацій проводити такі заняття, через неможливість проведення таких заходів постійно і, що найважливіше, складність повторення певного сценарію навіть для декількох студентів, а, як відомо, таких груп студентів здебільшого є декілька і т. ін.

Використання ділових ігор (Belchikov & Birshtejn, 1989; Vedenina, 2013), де ролі організації-замовника і фахівця-аналітика могли б виконувати охочі до навчання студенти, також некоректне, оскільки в основі визначення вимог знаходиться взаємодія двох (або більше) осіб з різних областей інтересів, знань і вмінь. Звичайно, в ролі замовника могли б виступати викладачі, проте їхня кваліфікація часто не зовсім адекватно відповідає реаліям сучасного виробництва. Водночас, вони володіють певними знаннями щодо визначення та аналізу вимог до ПЗ, а це суперечить умові на різні області інтересів. Хоча в роботі (Kungurcev, Blazhko & Marulin, 2010) запропоновано методику навчання основам проектування архітектури ПЗ з використанням рольових ігор, проте в ній не розглянуто допустимі варіанти реалізації етапів цього процесу, що заставляє викладачів (не фахівців у різних предметних областях) часто придумувати не зовсім реалістичні сценарії.

Цитування за ДСТУ: Грицюк Ю. І., Лешкевич І. Ф. Особливості визначення вимог до програмного забезпечення та проблеми їх аналізу. Науковий вісник НЛТУ України. 2017. Вип. 27(4). С. 148–158.

Citation APA: Grytsiuk, 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

Варто також згадати деякі проблеми, які часто виникають перед аналітиками-початківцями під час визначення вимог до ПЗ з представниками від організації-замовника, подальшого їхнього аналізу та узгодження сформульованих вимог з ними. Через відсутність тривалої практики використання аналітиків вітчизняними компаніями-розробниками ПЗ ще й на сьогодні не сформовано адекватний перелік видів діяльності аналітиків під час визначення та аналізу вимог до ПЗ. Водночас, часто аналітики при роботі з представниками від організації-замовника забувають чи не хочуть доводити до їхнього відома показники якості, яким має відповідати майбутнє ПЗ, знімаючи тим самим з себе певну відповідальність за виконану роботу. Понад це, у доступних літературних джерелах відсутні адекватні формалізовані моделі відповідних процедур, які стосуються процесу визначення та аналізу вимог до ПЗ.

Усе це зумовлює потребу додаткового вивчення зазначеного вище переліку проблем, які стосуються процесу визначення та аналізу вимог до ПЗ, що й становить актуальність цього дослідження.

Об'єкт дослідження – визначення та аналіз вимог до ПЗ.

Предмет дослідження – системний аналіз предметної області, що забезпечить реалізацію процедур визначення та аналізу вимог до ПЗ, а також дасть змогу формалізувати відповідні дії аналітика, які ефективно реалізують цей процес.

Мета дослідження полягає в проведенні системного аналізу предметної області, результати якого дадуть змогу достовірно описати нагальні проблеми визначення та аналізу вимог до ПЗ, з'ясувати види діяльності аналітика для ефективного вирішення цих проблем, а також запропонувати основні показники якості майбутнього ПЗ, розробити адекватні формалізовані моделі відповідних процедур визначення та аналізу вимог до ПЗ.

Для реалізації цієї мети потрібно виконати такі основні завдання:

1) з'ясувати нагальні проблеми, які стосуються процедури визначення вимог до ПЗ, встановити шляхи їх уникнення чи пом'якшення;
2) охарактеризувати дії аналітиків під час аналізу вимог до ПЗ,
3) запропонувати представникам від організації-замовника основні показники якості ПЗ, яким має відповідати майбутнє ПЗ;
4) удосконалити наявні формалізовані моделі відповідних процедур, які стосуються визначення та аналізу вимог до ПЗ;
5) зробити відповідні теоретичні висновки та надати рекомендації щодо практичного їх використання.

1. Нагальні проблеми визначення вимог до програмного забезпечення. *Розроблення ПЗ* (англ. *Software Development*) – це дії компанії-виконавця, які містять інженерію вимог до процесів проектування, кодування та тестування ПЗ з дотриманням правил інженерії його якості. Така діяльність компанії-розробника ПЗ вимагає від аналітика детального вивчення предметної області (Bobalo, et al., 2013), яка стосується безпосередньої діяльності організації-замовника, та повного розуміння цілей використання ними майбутнього програмного продукту (Mishhenko, Pomorova & Govorushhenko, 2012). На сьогодні відома велика кількість технологій реалізації проектів з розроблення ПЗ (Lavrishheva & Petruhin, 2006; McConnell, 2013). Незалежно від підходу, який використовується в тій чи іншій технології розроблення ПЗ, на початковому етапі життєвого циклу (ЖЦ) будь-якого програмного проекту аналітик має провести ґрунтовний аналіз проблемної області, забезпечити формування достатнього набору вимог до майбутнього програмного продукту та здійснити їх узгодження з замовником. Саме на цьому етапі мають бути зафіксовані реальні потреби потенційних користувачів ПЗ, що стосуються функціональних, операційних і сервісних його можливостей, які має реалізувати компанія-розробник. Правильно сформовані аналітиком набори вимог до ПЗ є гарантією того, що програмний продукт буде задовольняти усім побажанням зацікавлених сторін під час його використання (Gilb, 1988).

Відомо (McConnell, 2013), що сучасне ПЗ супроводжують такі основні проблеми його розроблення та використання:

- складність опису його ієрархічної структури і відповідних компонент, які стосуються кожної ієрархії;
- наявність множини тісно взаємопов'язаних компонент з локальними завданнями та цілями його використання;
- відсутність певних аналогів і, як наслідок, часта неможливість використання будь-яких типових проектних рішень;
- потреба інтеграції наявного і розробленого ПЗ у єдину корпоративну мережу, узгодження їхніх інтерфейсів;
- потреба функціонування ПЗ у неоднорідному середовищі на різних апаратних платформах як ближніх, так і віддалених користувачів;
- відокремленість та різнорідність окремих груп розробників ПЗ із різним рівнем кваліфікації його працівників;
- великі терміни реалізації програмного проекту через часті зміни зацікавленими сторонами вимог до ПЗ.

Програмний проект (англ. Software Project) – це комплекс взаємопов'язаних заходів, спрямованих на досягнення поставлених завдань з чітко визначеними цілями протягом заданого періоду часу та при встановленому бюджеті (Gilb, 1988). Такі проекти часто зазнають невдач через помилки на ранніх етапах життєвого циклу ПЗ, а саме (Pomorova & Hovorushchenko, 2013):

1) неадекватне формулювання вимог користувача до ПЗ чи неузгодженість їх з організацією-замовником;
2) невдале проектування архітектури ПЗ або неефективне планування етапів реалізації програмного проекту;
3) неправильне розуміння побажань замовника щодо можливостей майбутнього програмного продукту або недостатній аналіз специфікації вимог до ПЗ його розробниками, а також недостатня деталізація етапів реалізації програмного проекту;
4) нереалістичні проектні плани щодо вартості та тривалості реалізації програмного проекту, щодо ефективності етапів реалізації програмного проекту, щодо тривалості життєвого циклу ПЗ, щодо простоти та зручності використання майбутнього програмного продукту та його кросплатформенності;
5) неправильно підібрані члени команди-розробника ПЗ, недостатня кваліфікація керівника програмного проекту.

Вимоги до ПЗ **(**англ. *Software Requirement*) – набір характеристик щодо властивостей, якості та функцій майбутнього програмного продукту, який потрібно розробити або знаходиться у процесі модернізації (Vigers, 2004). Часто вимоги до ПЗ аналітики визначають під

час аналізу технічного завдання на його розроблення та фіксують у відповідній специфікації вимог, діаграмах прецедентів й інших артефактах процедури аналізу предметної області (Bobalo, et al., 2013).

За ієрархічними рівнями розрізняють такі види вимог до ПЗ (Lavrishheva & Petruhin, 2006):

- *бізнес-вимоги* – встановлюють призначення ПЗ, їх детально описують у документі про *бачення* (англ. *Vision*) та документі про *межі проекту* (англ. *Scope*);
- *вимоги користувача* – встановлюють набір завдань користувача, які має реалізувати готовий програмний продукт, а також сценарії їхнього вирішення в програмно-апаратній системі. Ці вимоги можуть мати вигляд тверджень, варіантів використання (англ. *Use Case*), історій користувача (англ. *User Stories*), сценаріїв взаємодії (англ. *Scenario*) тощо;
- *функціональні вимоги* – встановлюють "*що*" має робити готовий програмний продукт у програмно-апаратній системі, їх детально описують у специфікації вимог до ПЗ (англ. *Software Requirements Specification, SRS*).

Специфікація вимог до ПЗ (англ. *Software Requirements Specification, SRS*) – повний опис поведінки ПЗ, що потрібно розробити, це основа, яка містить множину функціональних і нефункціональних (додаткових) вимог до майбутнього програмного продукту. Функціональні вимоги описують всі взаємодії користувача з ПЗ, а нефункціональні вимоги – накладають обмеження на реалізацію програмного проекту чи можливості функціонування ПЗ (IEEE 830-1998, 1998).

Помилки, внесені під час формування набору вимог до ПЗ чи проектування його архітектури, становлять 25-55% від усіх помилок, причому чим більший обсяг ПЗ, тим більше помилок вноситься саме на ранніх стадіях його ЖЦ (Pomorova & Hovorushchenko, 2013). Вважається (Boegh, et al., 1999), що вартість виправлення помилки проектування архітектури ПЗ у 2-4 рази вища від вартості виправлення помилки під час його конструювання. Залежність вартості виправлення помилок від етапу ЖЦ ПЗ наведено у роботі (IEEE 1031-2011, 2008).

Процес визначення вимог до ПЗ поділяють на декілька етапів (Lavrishheva & Petruhin, 2006):

- *виявлення вимог* – збирання аналітиком побажань замовника, його бачень роботи та меж застосування майбутнього програмного продукту, встановлення потреб зацікавлених сторін і загальних можливостей програмно-апаратної системи;
- *аналіз вимог до ПЗ* – перевірка цілісності та завершеності сформованого набору вимог до програмного продукту, узгодження їх з представниками від зацікавлених сторін, внесення та відстеження змін;
- *специфікація вимог до ПЗ* – документування вимог до програмного продукту згідно з наявними стандартами та нормативними документами;
- *тестування вимог до ПЗ* – встановлення відхилень розробленого ПЗ від зазначеного набору вимог до програмного продукту (Yakovyna, et al., 2010).

Для виявлення вимог, яким має відповідати майбутній програмний продукт, використовують такі методи (Kungurcev, Kalinina & Novikova, 2013):

- спілкування з майбутніми користувачами: інтерв'ю, опитування, анкетування (Lukina, 2003);
- мозковий штурм, семінар, відео-конференція (Leffingujell, & Uidrig, 2002);
- спостереження за виробничою діяльністю користувачів майбутнього програмного продукту, "фотографування" їхнього робочого дня;
- аналіз нормативної документації та чинного законодавства (IEEE 1031-2011, 2008; IEEE 1061-26157, 1998; IEEE 830-1998, 1998);
- аналіз бізнес-процесів: моделей діяльності, конкурентних програмних продуктів, статистики використання попередніх версій ПЗ (ISO/IEC, ISO/IEC 25000, 2005; McCall, Richards, & Walters, 1977).

Відомі такі джерела, які допомагають аналітику здійснювати аналіз вимог до ПЗ (Lavrishheva & Petruhin, 2006): законодавство – чинні укази та постанови; вимоги стандартів; бізнес-процеси; очікування користувачів від роботи ПЗ та їхнє бачення щодо функціональних можливостей майбутнього програмного продукту.

Різні методології розроблення ПЗ можуть по-різному організовувати роботу аналітика щодо процедур визначення та аналізу вимог до ПЗ (Lavrishheva & Petruhin, 2006). У дуже старій та не зовсім актуальній на сьогодні моделі *водоспаду* (англ. *Waterfall*) процедури визначення та аналізу вимог до ПЗ є початковими (Leffingujell & Uidrig, 2002). Особливість цієї моделі полягає в тому, що ці процедури мають повністю завершитися ще до початку етапу проектування архітектури ПЗ та етапу його розроблення, тому ці етапи не можуть початися раніше завершення процедури аналізу вимог.

В ітеративних методологіях розроблення ПЗ процедури визначення та аналізу вимог до нього в різному обсязі є на кожній ітерації (Gilb, 1988).

2. Проблеми аналізу вимог до програмного забезпечення. Як було зазначено вище, процедура визначення вимог до ПЗ полягає у з'ясуванні потреб виконання та встановлення умов функціонування майбутнього програмного продукту, які висуваються різними представниками від організації-замовника. Водночас, процедура аналізу вимог до ПЗ зводиться до врахування можливих суперечностей між різними вимогами та вирішення різних конфліктних ситуації між замовниками та розробниками ПЗ, а також потенційними його користувачами чи, наприклад, його бенефіціарами1.

Вимоги користувачів до ПЗ подають у вигляді тверджень про їхні потреби, виражені користувачем чи групою осіб, зацікавлених у якісному створенні майбутнього програмного продукту та в успішному його використанні. Вони стосуються багатьох особливостей функціонування ПЗ, зокрема і поведінкових. Загалом, поведінкові вимоги користувачів до ПЗ можна поділити на два типи:

- функціональні вимоги – які встановлюють, що саме має бути реалізовано в ПЗ;
- вимоги до програмної продуктивності – які встановлюють, наскільки добре це має бути виконано.

Функціональні вимоги (англ. *Functional Requirements*) – вимоги до ПЗ, які описують внутрішню роботу програмного продукту, його поведінку: калькулювання даних, маніпулювання даними, оброблення даних й інші специфічні функції, які він має виконувати (Gilb, 1988). На відміну від нефункціональних вимог, які вста-

[1] Бенефіціар (від фр. benefice – прибуток, користь) – одержувач визначених вигод, що виникають внаслідок реалізації програмного проекту.

новлюють, яким програмний продукт має бути, функціональні вимоги встановлюють, що продукт має робити. Функціональні вимоги до ПЗ аналітик має з'ясувати на першій стадії процесу розроблення ПЗ, тобто на етапі аналізу вимог.

Нефункціональні вимоги (англ. *Non-Functional Requirements*) – вимоги до ПЗ, які задають критерії оцінювання якості його роботи (McCall, Richards & Walters, 1977). На відміну від функціональних вимог, які встановлюють, що програмний продукт має робити, нефункціональні вимоги встановлюють, яким продукт має бути. Нефункціональні вимоги аналітик має з'ясувати також на етапі аналізу вимог до ПЗ.

За призначенням нефункціональні вимоги до ПЗ спрямовані на покращення безпеки ПЗ, його надійності, швидкодії, зручності у використанні тощо, а також на вдосконалення (масштабування, відновлюваність ...) властивостей ПЗ.

Практики розрізняють такі види нефункціональних вимог до ПЗ:

1) Вимоги до інтерфейсу (*Interface Requirements*) поділяють на:
   - апаратні інтерфейси (*Hardware Interfaces*) – необхідні для підтримки роботи програмно-апаратної системи, зокрема її логічної структури, фізичних адрес і очікуваної поведінки;
   - інтерфейси ПЗ (*Software Interfaces*) – інтерфейси, з якими наявна аплікація має взаємодіяти з користувачем під час її використання;
   - комунікаційні інтерфейси (*Communications Interfaces*) – інтерфейси для комунікацій (взаємодії) ПЗ з іншими програмно-апаратними системами та/або пристроями.
2) Апаратні та програмні вимоги (*Hardware/Software Requirements*) – опис апаратної та програмної платформ, необхідних для надійної роботи (і підтримки) програмно-апаратної системи.
3) Операційні вимоги (*Operational Requirements*) відповідають за:
   - безпеку та конфіденційність (*Security and Privacy*) даних у ПЗ;
   - надійність (*Reliability*) роботи ПЗ у штатних і позаштатних ситуаціях;
   - відновлюваність (*Recoverability*) ПЗ після аварійного завершення роботи;
   - програмну продуктивність (*Performance*) ПЗ у різних наборах вхідних даних;
   - потенціал (місткість) (*Capacity*) ПЗ стосовно обсягу оброблюваних даних;
   - збереження даних (*Data Retention*) в оперативній пам'яті ПЗ;
   - управління помилками (*Error Handling*) в ПЗ, які виникають внаслідок його роботи, але не були виявлені під час тестування (Futrell, Shafer, & Shafer, 2003);
   - правила перевірки (*Validation Rules*) функціональних можливостей ПЗ;
   - узгоджені стандарти (*Convention Standards*), згідно з якими було розроблено ПЗ.

Отже, усі функціональні та нефункціональні вимоги до ПЗ підлягають детальному аналізу, узгодженню з представниками від замовників і розробниками програмного продукту. Здійснення такого аналізу вимог до ПЗ є критичним етапом реалізації програмного проекту, від якісного виконання якого залежить успішне його завершення (Kobern, 2011). Під час такого аналізу всі вимоги мають бути задокументованими, виміряними, протестованими (Yakovyna, et al., 2010), пов'язаними з бізнес-потребами майбутніх користувачів ПЗ, а також описані з рівнем деталізації, достатнім для проектування архітектури ПЗ. Аналіз вимог до ПЗ аналітик має здійснювати за *архітектурними*, *структурними*, *поведінковими*, *функціональними* та *експлуатаційними* характеристиками розроблюваного нового чи модифікованого наявного програмного продукту.

Архітектура ПЗ (англ. *Software Architecture*) – структура програми або обчислювальної системи, яка містить програмні компоненти, видимі ззовні властивості цих компонент, а також відносини між ними. Якісний аналіз вимог до архітектури ПЗ з подальшим його документуванням значно спрощує відносини між зацікавленими сторонами (англ. *Stakeholders*) та розробником ПЗ, дає змогу зафіксувати рішення, прийняті на ранніх етапах реалізації програмного проекту, узгодити його якісний дизайн, а також уможливлює використання компонент цього дизайну і шаблонів проектування повторно в інших програмних проектах (McConnell, 2013).

Проектування архітектури ПЗ – процес розроблення його ієрархічної структури, що виконується після процедур визначення й аналізу вимог до ПЗ. Завдання такого проектування – перетворення сформованого набору вимоги до ПЗ у архітектуру ПЗ (Leffingujell & Uidrig, 2002). Таку побудову аналітик має здійснювати шляхом визначення цілей діяльності реальної системи, її вхідних і вихідних даних, декомпозиції системи на підсистеми, на компоненти або модулі та об'єднання їх у загальну структуру. Проектування архітектури ПЗ може виконуватися різними методами (стандартизованим, об'єктно-орієнтованим, компонентним й ін.), кожний з яких передбачає свій перелік виконання робіт, а саме – визначення концептуальної, об'єктної й інших моделей за допомогою відповідних конструктивних елементів (блок-схем, графів, структурних діаграм тощо).

Аналіз вимог до структури окремих програмних компонент і організації файлової системи для міжпрограмного обміну даними стосується, в основному, підготовки програмної документації до ПЗ. Розрізняють *зовнішню програмну документацію*, яка узгоджується із замовником, і проміжну – *внутрішню документацію* до програмного проекту.

Під час підготовки програмної документації аналітик має спочатку розробити зовнішні специфікації вимог, а потім – внутрішні. *Зовнішні специфікації вимог* містять перелік вхідних і вихідних даних, їх структурну організацію (статичну чи динамічну), реакції ПЗ на виняткові ситуації, визначеність у тому, що саме має робити користувач (за якими алгоритмами він має діяти і звідки має брати інформацію), а що має виконувати програмно-апаратна система. Тобто зовнішні специфікації вимог мають містити все те, що побачив би користувач, коли б він отримав готовий програмний продукт. Ефективність аналізу зовнішньої специфікації вимог до ПЗ залежать від тривалості та стадій його життєвого циклу.

Під час розроблення ієрархічної структури ПЗ, перед початком реалізації програмного проекту, до процесу тестування зовнішніх специфікацій вимог варто залучати потенційних користувачів майбутнього програмного продукту. Користувачам також варто показувати макети екранів у порядку виконання програмою усіх своїх функцій, після чого вони зможуть готувати дані для їх тестування, а також згодом зможуть без зайвих зусиль апробувати та відлагодити методику роботи з усім ПЗ загалом.

Внутрішні специфікації вимог містять опис внутрішніх даних програми (змінних, особливо структурованих) і опис алгоритмів роботи усієї програми та окремих її модулів чи частин. Такі специфікації подають у поєднанні з описом архітектури ПЗ і внутрішньої структури окремих його компонент.

3. Дії аналітика під час аналізу вимог до програмного забезпечення. Під час аналізу вимог до ПЗ аналітик має виконувати такі види діяльності (Futrell, Shafer & Shafer, 2003):

- *виявлення/збирання потреб* – завдання комунікації аналітика з представниками від замовника, зокрема з користувачами майбутнього програмного продукту, для з'ясування їхніх потреб та узгодження сформованого набору вимог до нього;
- *запис вимог* – документування вимог до ПЗ у різних формах, таких як опис звичайною мовою, у вигляді прецедентів, користувацьких історій діяльності чи традиційних специфікацій вимог;
- *аналіз вимог до ПЗ* – виявлення недоліків вимог до ПЗ (неточностей, неповноти, неоднозначностей чи суперечностей, конфліктних ситуацій) і їх виправлення, а також перевірка можливості, тривалості та вартості реалізації.

Відомо (Blagodatskih, Volnin & Poskakalov, 2005), що процедура аналізу вимог до ПЗ може бути тривалою та ресурсовитратною, що вимагає від аналітиків знання деталей предметної області та застосування тонких психологічних навиків. Оскільки нове ПЗ змінює середовище його використання та відношення між користувачами, аналітикам важливо розпізнати характерні особливості всіх зацікавлених сторін, взяти до уваги всі їхні потреби і переконатись в тому, що вони розуміють наслідки, які привносить нова програмна система в бізнес-процеси.

Зазвичай, аналітики можуть використати декілька методів, щоб отримати вимоги до ПЗ від кінцевих її користувачів (Vigers, 2004; Kobern, 2011; McConnell, 2013). Однак, історично склалося так, що ця процедура містить проведення інтерв'ю чи організацію фокус-груп (яку в цьому контексті частіше називають як майстерня вимог), де потреби представників від зацікавлених сторін ретельно записують, після чого ці записи обробляють відповідно до наявних методик і створюють так звані набори вимог користувачів.

До сучасніших підходів виявлення/збирання вимог до ПЗ належать прототипування та підготовка прецедентів (Leffingujell & Uidrig, 2002). За потреби, аналітики можуть використовувати комбінацію цих методів, щоб встановити точні потреби зацікавлених сторін, тобто розробити їх так, щоб ПЗ під час його використання повністю відповідало зазначеним бізнес-потребам.

У роботі (Lavrishheva & Petruhin, 2006) виокремлено такі основні етапи процедури аналізу вимог до ПЗ:

1) Визначення зацікавлених сторін у розробленні ПЗ. З'ясування переліку зацікавлених сторін, встановлення їхніх потреб та інтенсивності (частоти) використання майбутнього програмного продукту часто застосовується для опису можливостей його застосування в інших бізнес-процесах. При цьому зацікавлені сторони – особи чи організації, які дійсно претендують на придбання програмного продукту, який може безпосередньо чи опосередковано впливати на їхню бізнес-діяльність. Багато практиків вважають, що зацікавлені сторони не завжди обмежуються однією компанією-розробником ПЗ, яка присилає їм своїх аналітиків для встановлення сценаріїв використання чи запису користувацьких історій діяльності.

До зацікавлених сторін можуть належати такі представники (Lavrishheva & Petruhin, 2006):

- ті, хто управлятиме ПЗ – звичайні та обслуговуючі оператори;
- ті, хто отримає вигоди від ПЗ – функціональні, фінансові, політичні та соціальні бенефіціари;
- ті, хто бере участь у фінансуванні процесу розроблення ПЗ, його придбанні чи закупівлі. Якщо програмний продукт розробляють для масового ринку, то відділ менеджменту та маркетингу, а іноді й відділ продажу діють як сурогатні його споживачі, які спрямовують процес його розроблення в правильному руслі;
- організації, що регулюють напрями функціонування ПЗ – фінансові, безпекові та регуляторні;
- особи та організації, що проти розроблення ПЗ – негативні зацікавлені сторони (негативний прецедент);
- особи та організації, що відповідають за розроблення ПЗ, які будуть взаємодіяти з ним;
- ті організації, які горизонтально інтегруються із організацією, для якої аналітики конструюють ПЗ.

2) Інтерв'ю із зацікавленими сторонами щодо їхніх потреб від майбутнього ПЗ. Проведення інтерв'ю із представниками від зацікавлених сторін є традиційним підходом, що використовується під час аналізу вимог до ПЗ. Таке уточнення вимог слугує одним із шляхів отримання цілісного та достовірного знання, яке часто неможливо виявити в спільних сесіях із замовниками і розробниками ПЗ, де їхня увага підпорядкована забезпеченню більш крос-функціонального контексту. Понад це, особистий характер проведення інтерв'ю із конкретними респондентами надає аналітику більш довірливе середовище для спілкування, де прогалини в його знаннях можна дещо детальніше пояснити.

3) Спільні сесії визначення вимог до ПЗ. Вимоги до ПЗ часто мають крос-функціональні наслідки, які не завжди відомі окремим представникам від зацікавлених сторін, тому їх часто пропускають чи неправильно описують під час проведення інтерв'ю із респондентами. Такі негативні наслідки можна виявити шляхом проведення сесій між зацікавленими сторонами у контрольованому середовищі, роботу якого стимулює кваліфікований посередник – модератор[1]. При цьому зацікавлені сторони беруть участь у дискусії, внаслідок проведення якої вдається виявити деякі крос-функціональні вимоги до ПЗ, проаналізувати їх деталі та, що найважливіше, розкрити їх крос-функціональні наслідки. При цьому мають бути присутні спеціально призначені секретар і бізнес-аналітик, в обов'язки яких входить документування самої дискусії та аналіз її конкретних моментів відповідно. Використання ж навиків навченого посередника (модератора) для управління дискусією звільняє бізнес-аналітика від зайвої роботи, даючи йому змогу зосередити свої думки на процесі виявлення крос-функціональних вимог до ПЗ (Futrell, Shafer & Shafer, 2003).

Сесії між представниками від зацікавлених сторін подібні до сесій спільного проектування архітектури ПЗ, які спершу виявляють вимоги до архітектури ПЗ, а

[1] Модератор (з лат. той, що стримує) – людина, що проводить соціологічні дослідження; ведучий фокус-груп.

потім формалізують його властивості, які мають бути реалізовані в майбутньому ПЗ для того, щоб задовольнити потреби його користувачів.

4) Набори вимог до ПЗ у стилі контракту. Традиційним способом документування вимог до ПЗ є формування їх переліку в стилі контракту. В складному ПЗ такий набір вимог може розтягуватись на сотні сторінок. Відповідним аналогом може слугувати надзвичайно великий асортимент продукції в супермаркеті. Однак, набори вимог до ПЗ у стилі контракту не дуже цінуються аналітиками під час їх аналізу, оскільки вони показують малу ефективність у досягненні своїх бізнес-цілей. Проте, такі вимоги трапляються ще й сьогодні, особливо тоді, коли замовник ПЗ має гіркий досвід прийняття в експлуатацію програмного продукту, що не зовсім відповідав їхнім вимогам.

Документування вимог до ПЗ у стилі контракту мають такі переваги:

- контракт надає чіткий попунктовий набір вимог до ПЗ, конкретне формулювання та однозначне трактування;
- надає контракт між замовниками (спонсорами) програмного проекту та його розробниками;
- для складного ПЗ може надати детальний опис вимог до нього.

Недоліки контракту вимог до ПЗ:

- такі набори вимог до ПЗ розтягуються на сотні сторінок, внаслідок чого ці вимоги майже неможливо повністю осмислити й отримати повне розуміння роботи ПЗ;
- такі набори абстрагують всі вимоги до ПЗ, тому в них є мало контексту:
  - абстракція вимог робить неможливим бачення того, як саме вони поєднуються між собою чи взаємодіють разом;
  - абстракція вимог заважає правильно розставляти пріоритети між ними;
  - абстракція вимог збільшує їх подібність та ймовірність неправильної їх інтерпретації: чим більше фахівців їх осмислять, тим більша кількість різних інтерпретацій з'явиться;
  - абстракція вимог заважає розробникам ПЗ впевнитись в тому, що вони мають у наявності більшість вимог, які потрібно врахувати в готовому програмному продукті;
- такі набори вимог створюють фальшиве відчуття взаємного порозуміння між замовниками та розробниками ПЗ;
- набори вимог у стилі контракту дають зацікавленим сторонам фальшиве відчуття безпеки в тому, що розробники ПЗ змушені будуть досягти тих висот, які вони замовили. Однак, через свою природу такі набори вимог часто упускають критичні вимоги, які проявляються пізніше в процесі розроблення ПЗ. Також розробники ПЗ часто використовують такі відкриті вимоги для того, щоб переглянути умови договору на свою користь;
- такі набори вимог не зовсім допомагають розробникам ПЗ у ефективному його проектуванні.

Як альтернатива до великих, попередньо описаних наборів вимог до ПЗ, є гнучке розроблення ПЗ, яке використовує історії користувачів для опису вимог про їхню діяльність у повсякденних термінах.

5) Вимірювані цілі застосування майбутнього ПЗ. Найкращі практики (проектанти та розробники ПЗ) беруть сформований набір вимог як підказку до процедури проектування архітектури та власне розроблення ПЗ. При цьому вони можуть стосовно предметної області запитувати замовника "чому це відбувається так, а не інакше?" доти, доки не стане зрозумілою справжня бізнес-ціль програмного продукту. Замовники ПЗ і його розробники можуть скласти спеціальні тести, які дають їм змогу виміряти рівень досягнення поставлених цілей. Зазвичай такі цілі змінюються повільніше, ніж довгий перелік конкретних, але невимірних вимог. Як тільки невеликий набір критичних, але вимірних цілей буде встановлено, то швидке прототипування та короткі ітеративні етапи розроблення ПЗ часто можуть приносити значно більшу користь для замовника ПЗ, інколи навіть задовго до середини реалізації програмного проекту.

6) Прототипи – макети майбутнього ПЗ. Усередині 1980-их технологію прототипування ПЗ розглядалось як найкраще рішення до вирішення проблеми аналізу вимог до ПЗ. Прототипи ПЗ, тобто його макети, дають змогу розробникам ПЗ візуалізувати ще не створений програмний продукт. Водночас, прототипи допомагають майбутнім користувачам ПЗ уявити, як саме буде виглядати готовий програмний продукт, а також дає змогу розробникам ПЗ значно спростити процедуру прийняття архітектурних рішень. Свого часу після введення прототипів спостерігалося значне покращення комунікації між замовниками ПЗ і його розробниками. Раннє бачення користувачів майбутніх можливостей ПЗ дає змогу його замовникам значно зменшити кількість змін у подальшому його використанні, що призводить до зменшення загальної вартості розроблення ПЗ.

Однак, протягом останнього десятиліття прототипування, яке себе зарекомендувало як корисна методика аналізу вимог до ПЗ, не повністю вирішило проблему визначення та аналізу вимог до ПЗ, а саме:

- як тільки замовники бачать прототип ПЗ, то вони перестають розуміти, чому розробники ПЗ не зможуть завершити програмний проект упродовж деякого часу;
- розробники ПЗ часто думають, що змушені будуть використовувати склеєний докупи код прототипу в реальній ПЗ, боячись "втратити час" почати заново;
- загалом, прототипи допомагають розробникам ПЗ у прийнятті архітектурних рішень щодо проектування інтерфейсу користувача. Однак, прототипи не завжди можуть вказати їм, які вимоги були спочатку в замовника ПЗ;
- розробники ПЗ і кінцеві його користувачі можуть занадто довго зосередитись на побудові відповідних інтерфейсів, і занадто мало – на розробленні власне ПЗ, яка має виконувати певні бізнес-процеси;
- прототипи придатні для розроблення інтерфейсів користувача, які відображають реальні події на екрані, але не дуже корисні для реалізації пакетних чи асинхронних процесів, які можуть містити складні розрахунки чи цілеспрямовані запити для пошуку даних.

Прототипи деяких елементів ПЗ можна подати у вигляді плоских діаграм (які часто називають каркасними моделями) чи робочих програмних додатків, що використовують синтезовану функціональність. Каркасні моделі створюють за допомогою різноманітних графічних дизайн-документів, тому розробники ПЗ часто видаляють увесь колір з дизайну (використовують чорно-білу палітру) для того, щоб кінцеве ПЗ мало дещо безликий графічний дизайн. Це допомагає розробникам ПЗ безпосередньо уникнути плутанини між концептом програми в дизайн-документі та її кінцевим виглядом.

7) Прецеденти – технологія документування вимог до ПЗ. Прецеденти дають змогу задокументувати потенційні вимоги як до нового, так і для модифікованого ПЗ. Кожен прецедент надає один чи більше *сценаріїв*, які виражають те, як ПЗ має взаємодіяти з користувачем чи іншим ПЗ, щоб досягти конкретної бізнес-цілі.

Прецеденти, зазвичай, уникають технічного жаргону, віддаючи перевагу мові кінцевого користувача чи досвідченого експерта в предметній області. Інженерія вимог, як методологія розроблення ПЗ, та зацікавлені сторони в успішній його реалізації часто виступають співавторами прецедентів.

Для початківців часто прецеденти видаються оманливо простими інструментами для опису поведінки ПЗ. Хоча прецедент і містить текстовий опис всіх способів, якими користувачі можуть працювати з ПЗ, однак вони не описують жодних внутрішніх процесів у ньому, а також не пояснюють, як саме ПЗ має бути реалізоване. Вони просто показують кроки, які користувач має здійснити, щоб виконати своє завдання. За аналогією можна описати й усі способи взаємодії користувачів з майбутнім ПЗ.

8) Специфікація вимог до ПЗ (англ. *Software Requirements Specification, SRS*) – повний опис поведінки ПЗ, яке потрібно розробити, містить набір прецедентів, які описують всі взаємодії користувача з ПЗ. Прецеденти також відомі як функціональні вимоги до ПЗ. Окрім прецедентів, SRS також містить нефункціональні (додаткові) вимоги, які накладають обмеження на програмний проект чи його реалізацію (такі як вимоги певної продуктивності ПЗ, дотримання стандартів якості чи обмеження на проектування).

Рекомендовані підходи для підготовки специфікації вимог до ПЗ описано в стандарті IEEE 830–19981, в якому зазначено всі можливі структури, бажану структуру та вміст її елементів, а також якість програмного продукту загалом.

Валідація вимог – це перевірка вимог до ПЗ для переконання в тому, що вони визначають саме дану виробничу систему. Замовник проводить експертизу зафіксованого набору вимог для того, щоб розробник майбутнього ПЗ зміг далі виконувати проектування його архітектури. Один з методів валідації – прототипування, тобто швидке відпрацювання окремих вимог на конкретному інструменті, аналіз масштабу його виконання та потенційні зміни вимог, вимірювання функціональності та вартості майбутнього програмного продукту, а також визначення зрілості процесів визначення вимог.

Верифікація вимог – це процес перевірки правильності специфікації вимог до ПЗ на їхню відповідність стандартам і функціям програмної системи. Внаслідок перевірки вимог створюється остаточний і погоджений документ, що встановлює повноту і коректність вимог до ПЗ, а також можливість продовжити проектування його архітектури.

4. Показники якості програмного забезпечення. ***Якість ПЗ*** – сукупність властивостей програмного продукту, що встановлюють його спроможність задовольнити запити замовника, які він висловив у вигляді користувацьких вимог на початкових етапах розроблення ПЗ (McConnell, 2013). Згідно з міжнародними та вітчизняними стандартами оцінювання рівня якості програмного продукту, виділяють два процеси забезпечення якості ПЗ впродовж його життєвого циклу:

1) *гарантія якості ПЗ*, що є результатом певних дій на кожній стадії життєвого циклу програмному продукту з перевірки й підтвердження відповідності його стандартам і процедурам, орієнтованим на досягнення певних характеристик якості;
2) *інженерія якості ПЗ* як процес надання програмному продукту, який потрібно розробити або знаходиться у процесі розроблення, надійності, супроводу й інших характеристик якості.

Ці процеси забезпечення якості ПЗ потребують (Futrell, Shafer & Shafer, 2003):

- оцінювання стандартів і процедур, що використовуються під час розроблення ПЗ;
- ревізії процесів управління, розроблення та забезпечення якості ПЗ, а також усієї проектної документації (звітів, графіків розроблення, повідомлень);
- контролю за процесом проведення формальних інспекцій та оглядів певних дій на кожному етапі реалізації програмного проекту;
- аналізу й контролю за процесом проведення тестування (випробування) ПЗ.

Функціональність ПЗ – сукупність властивостей, які встановлюють спроможність програмного продукту виконувати в заданому середовищі упорядковану послідовність дій для задоволення споживчих властивостей, замовлених користувачем, відповідно до вимог оброблення даних і загальносистемних засобів. Функціональності ПЗ характеризується такими атрибутами:

- *функціональна повнота* – атрибут, який показує ступінь достатності основних функцій програми для вирішення спеціальних завдань згідно з її призначенням;
- *правильність* – атрибут, який показує, як забезпечується досягнення правильних і погоджених результатів оброблення даних внаслідок виконання програми;
- *інтероперабельність* або *сумісність* – атрибут, який вказує на спроможність програми взаємодіяти з іншими програмними системами й середовищами;
- *захищеність* – атрибут, який вказує на можливість запобігати несанкціонованому доступу до певних послідовностей дій, які виконуються програмою, а також до даних, які обробляються нею;
- *узгодженість* – атрибут, який вказує на відповідність програми чинним стандартам, угодам, правилам, законам і розпорядженням.

Надійність ПЗ – це: 1) множина атрибутів, які вказують на спроможність програмного продукту коректно перетворювати вхідні дані на очікувані результати; 2) сукупність властивостей, яка характеризує здатність програмного продукту безвідмовно виконувати певні функції та зберігати заданий рівень придатності до роботи при заданих умовах функціонування протягом певного інтервалу часу з достатньо великою ймовірністю. Надійність ПЗ прийнято характеризувати рівнем його завершеності (відсутністю помилок), стійкістю до появи помилок і можливістю перезапуску після аварійної зупинки. Зниження надійності ПЗ відбувається внаслідок не виявлення помилок у вимогах до ПЗ, у неправильному виконанні процесів його проектування та реалізації програмного проекту.

Надійності ПЗ характеризується такими атрибутами:

- *безвідмовність* – атрибут, який встановлює частоту відмов програми внаслідок наявності помилок у ній;
- *стійкість до помилок* – атрибут, який вказує на забезпечення спроможності програми виконувати свої функції в аномальних умовах (збої апаратури, помилки в даних і інтерфейсах, порушення в діях оператора тощо);

[1] IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications – Description

- *відновлюваність* – атрибут, який вказує на спроможність програми до її перезапуску для повторного виконання й відновлення даних після відмов у доступі до них;
- *узгодженість* – атрибут, який показує відповідність програми чинним стандартам, угодам, правилам, законам і розпорядженням.

Надійність ПЗ характеризується такими показниками (Blagodatskih, Volnin & Poskakalov, 2005):

1) *тривалість напрацювання на відмову* – вимірюється тривалістю роботоздатного стану ПЗ між двома послідовними відмовами або початком нормального функціонування ПЗ після них;
2) *тривалість відновлення* – враховує можливість багаторазових відмов і відновлень після аварійної зупинки програми;
3) *готовність до роботи* – відображає ймовірність мати відновлювану систему в роботоздатному стані в довільний момент часу. Значення показника відповідає частці тривалості корисної роботи ПЗ на достатньо великому інтервалі, який містить відмови і відновлення;
4) *напрацювання на ситуацію відмови або стійкість* – значення тривалості між втратами роботоздатності ПЗ незалежно від того, наскільки швидко відбулось його відновлення.

Для визначення кількісних значень показників надійності ПЗ використовують аналітичні та емпіричні моделі надійності ПЗ – математичні моделі, побудовані для оцінювання залежності надійності ПЗ від деяких певних параметрів (Blagodatskih, Volnin & Poskakalov, 2005). Наприклад, деякі типи програмних систем (реального часу, радарні, безпеки, комунікації, медичного устаткування тощо) містять особливі вимоги до забезпечення високої надійності з такими атрибутами, як *недопустимість помилок*, *забезпечення безпеки*, *захищеність і зручність застосування*, а також *достовірність результатів* роботи системи як основний критерій надійності.

Зручність застосування ПЗ – множина атрибутів, що характеризують умови взаємодії користувача із програмним продуктом, характеризується такими основними атрибутами:

- *зрозумілість* – атрибут, який вказує, наскільки зрозумілі користувачу для розпізнавання логічні концепції ПЗ та умови їх застосування;
- *легкість навчання* – атрибут, який вказує, наскільки доступні (легкі) для вивчення умови використання ПЗ;
- *оперативність* – атрибут, який характеризує швидкістю реакції ПЗ на дії користувача;
- *узгодженість* – атрибут, який показує відповідністю процесу розроблення ПЗ вимогам чинних стандартів, угод, правил, законів і розпоряджень;

Ефективність роботи ПЗ – зв'язок між результатами використання програмного продукту та кількістю задіяних для цього ресурсів (апаратних, матеріальних, праці обслуговувального персоналу тощо).

Супроводжуваність ПЗ – зусилля, які потрібно витратити на коригування, вдосконалення й адаптацію програмного продукту у разі зміни середовища його використання, вимог або функціональних специфікацій.

Супроводжуваність ПЗ характеризується такими показниками й атрибутами:

- *аналізованість* – показник, який встановлює потрібні зусилля для діагностики причин відмов програми або ідентифікації її частин, що потрібно модифікувати;
- *змінюваність* – показник, який встановлює потрібні зусилля на модифікацію програми, усунення помилок або внесення змін у зв'язку з виявленими помилками чи новими можливостями середовища функціонування;
- *стабільність* – атрибут, що характеризує ймовірність удосконалення та модифікації програми;
- *тестованість* – атрибут, що характеризує зусилля, потрібні для проведення валідації та верифікації програми.

Переносність ПЗ – здатність програмного продукту пристосовуватися до різних середовищ його розроблення та використання, забезпечуючи при цьому надійну та безперебійну роботу. До основних компонент середовища розроблення ПЗ належить: *організаційно-технологічне*, *апаратне*, *програмне*.

Атрибути переносності ПЗ є такими: *адаптивність*, *налагоджуваність*, *сумісність*, *узгодженість*, *інтероперабельність*.

Оцінювання якості ПЗ – дії, які мають визначити, якою мірою програмний продукт відповідає своєму призначенню.

5. Модель процесу визначення та аналізу вимог до ПЗ. Модель процесу визначення та аналізу вимог до ПЗ – це схема процесів його ЖЦ, що виконуються від початку зародження проекту майбутнього ПЗ і доти, доки не будуть визначені та сформовані набори вимог, проаналізовані та погоджені сформульовані вимоги із зацікавленими сторонами. Також у цій моделі мають бути передбачені процедури введення відповідних змін до вимог, введення цих змін у поточній версії ПЗ чи їх реструктуризація в наступних релізах.

Перед тим, як навести модель процесу визначення та аналізу вимог до ПЗ, спробуємо спочатку з'ясувати основні вимоги до такої моделі. Згідно з даними, наведеними в роботі (Kungurcev, Kalinina & Novikova, 2013), вимоги до моделі процесу визначення та аналізу вимог до ПЗ можуть мати такий вигляд:

1) Встановлення назви і короткої характеристики ПЗ, що дасть змогу аналітику сформувати стратегію і вибрати тактику реалізації подальших дій.
2) З'ясування структури діяльності організації-замовника в обсязі, достатньому для виявлення та отримання доступу до всіх зацікавлених осіб у розробленні ПЗ.
3) Подання способів з'ясування потреб до ПЗ, асоційованих із зацікавленими особами, формування набору вимог.
4) Подання документів і видів діяльності, асоційованих з робочими місцями в підрозділах організації-замовника.
5) Надання можливості аналітику відвідувати робочі місця у підрозділах організації-замовника, вибирати способи виявлення їхніх потреб, визначити і сформулювати вимоги до ПЗ, а також сформувати фрагменти відповідної специфікації вимог.
6) Надання аналітику підказок до різних методик виконання робіт та послідовності реалізації дій, пов'язаних із процесом визначення вимог, однак не пов'язаних з конкретною організацією-замовником.
7) Зберігання послідовності (історії) дій роботи аналітика для подальшого їх аналізу та оцінювання сформованого набору вимог до ПЗ.
8) Оцінювання результатів роботи аналітика шляхом аналізу його послідовності дій як експертною системою, так і реальними експертами і порівняння розробленої ним специфікації вимог з еталоном, наявним у системі.

Структура моделі експертної системи. Надалі будемо називати модель процесу визначення та аналізу

вимог до ПЗ його моделлю експертної системи. Її можна подати у вигляді кортежу, який складається із таких трьох моделей

$$M = \langle Mo, Mi, Ma \rangle, \quad (1)$$

де: *Mo* – модель організації-замовника (модель предметної області); *Mi* – інформаційна модель; *Ma* – модель роботи аналітика.

Модель організації-замовника призначена для подання її структури і, що найважливіше, робочих місць, для яких якраз і має розроблятися ПЗ. Модель імітує діяльність конкретних робочих місць та відповідних підрозділів організації, зв'язки їхнього підпорядкування та взаємодії, що існують у ній. Модель організації-замовника *Mo* подамо у вигляді множини вузлів

$$Mo = \{U_j,\ j = \overline{1,n}\}, \quad (2)$$

де *n* – кількість вузлів. Кожен вузол є деякою структурною одиницею організації-замовника, який подамо у вигляді такого кортежу:

$$\boldsymbol{U} = \left\{U_j = \langle N, W, H, S, C \rangle,\ j = \overline{1,n}\right\}, \quad (3)$$

де: *N* – назва (ідентифікатор) вузла; *W* – структурний підрозділ (відділ), до якого входить вузол; *H* – множина зв'язків "підлеглий – начальник"; *S* – множина зв'язків "начальник – підлеглий"; *C* – множина зв'язків співпраці між начальником і підлеглим.

Кожен зв'язок можна подати кортежем, наприклад

$$H = \langle K, Z \rangle, \quad (4)$$

де: $K = \{k_i,\ i = \overline{1,m}\}$ – вузол, з яким пов'язаний *N*-ий вузол; $Z = \{z_i,\ i = \overline{1,m}\}$ – стан зв'язку: доступний для переходу – true, закритий – false; *m* – кількість зв'язків.

Інформаційна модель слугує для подання інформації, яку аналітик-стажер може отримати на кожному робочому місці у будь-якому підрозділі організації-замовника, алгоритми управління процесом навчання, еталони вимог, які потрібно сформувати. Її можна подати у вигляді кортежу, який складається із трьох моделей

$$Mi = \langle R, P, D \rangle, \quad (5)$$

де: $R = \{R_j,\ j = \overline{1,Nv}\}$ – множина варіантів отримання інформації про вимоги, пов'язаних з кожним вузлом моделі організації; *P* – система допомоги; *D* – визначення документації, яку має сформувати аналітик-стажер; *Nv* – кількість варіантів отримання інформації про вимоги. Поточний *j*-ий варіант отримання інформації про вимоги R_j можна подати у вигляді такого кортежу

$$\boldsymbol{R} = \left\{R_j = \langle N_j, Fw, Vi \rangle,\ j = \overline{1,Nv}\right\}, \quad (6)$$

де: $N_j = \{n_{ji},\ i = \overline{1,m};\ j = \overline{1,Nv}\}$ – отримання *i*-ої інформації про вимоги із поточного *j*-го вузла; *Fw* – функція підтвердження можливих дій аналітика в поточному вузлі; $Vi = \{V_j,\ j = \overline{1,n}\}$ – множина варіантів отримання інформації аналітиком і можливості управління його подальшою поведінкою.

Подамо кожен варіант V_j у вигляді такого кортежу

$$\boldsymbol{V} = \left\{V_j = \langle Ver, Val, Fa, Fn \rangle,\ j = \overline{1,Nv}\right\}, \quad (7)$$

де: *Ver* – одна із множини версій отримання вимог (*Ver* ∈ *MVer*); *Val* – еталонний вибір; *Fa* – функція аналізу вибору аналітика та формування елемента вимог в його редакції; *Fn* – функція навігації, яка керує зв'язками між вузлами залежно від результатів діяльності аналітика.

В інформаційній моделі *Mi* організації-замовника використовуються такі варіанти отримання вимог від її зацікавлених осіб:

$$MVer = \langle Wi, Vuc, Vd, Va \rangle, \quad (8)$$

де: *Wi* – взяття інтерв'ю; *Vuc* – фіксування основних варіантів використання ПЗ; *Vd* – визначення документів, що беруть участь у документообігу організації-замовника; *Va* – прототипування.

Програмну документацію можна подати такими двома складниками

$$Doc = \langle Cont, Seq \rangle,$$

де: *Cont* – зміст специфікації вимог до ПЗ; *Seq* – містить допустиму послідовність заповнення розділів специфікації вимог до ПЗ.

Модель роботи аналітика слугує для:

- ідентифікації аналітика, який здійснює визначення та аналіз вимог до ПЗ;
- визначення вимог про ПЗ за версією аналітика;
- відновлення позиції аналітика після перерви в роботі з моделлю;
- оцінка роботи аналітика з моделлю, нагромадження статистичних даних про процес навчання.

Модель роботи аналітика можна подати у вигляді такого кортежу:

$$Mp = \langle Ns, Hs, Rs \rangle, \quad (9)$$

де: *Ns* – прізвище (ідентифікатор) аналітика; *Hs* – історія роботи аналітика з моделлю; *Rs* – результати роботи аналітика з моделлю (специфікація вимог).

Історію роботи аналітиків можна подати у вигляді такого кортежу:

$$Hs = \left\langle Ns_j, Val_j, Seq_j,\ j = \overline{1,Na} \right\rangle, \quad (10)$$

де: $Ns_j = \{n_{ji},\ i = \overline{1,m};\ j = \overline{1,Na}\}$ – множина вузлів, до яких отримав доступ *j*-ий аналітик; $Val_j = \{v_{ji},\ i = \overline{1,m};\ j = \overline{1,Na}\}$ – оцінки поточної діяльності *j*-го аналітика в *i*-му вузлі; $Seq_j = \{s_{ji},\ i = \overline{1,m};\ j = \overline{1,Na}\}$ – розділи еталона документації у *i*-му вузлі, які були доступні для *j*-го аналітика; *Na* – кількість потенційних аналітиків.

На початку роботи аналітика з експертною системою зміст специфікації вимог *Cont* копіюється в специфікацію вимог аналітика *Rs*. У процесі виконання роботи виявлені аналітиком вимоги до ПЗ заносяться до відповідних розділів *Rs*.

Удосконалена вище модель дає змогу для аналітика організувати процес визначення та аналізу вимог до ПЗ, який значною мірою наближається до реального. Розроблено структури документів, які не суперечать загальновизнаним вимогам, а також процедури їх використання та формування, що дає змогу формувати змістовні звіти за результатами навчання, контролювати процес навчання, фіксувати помилки й оцінювати знання та вміння аналітика. Запропоновану модель можна деталізувати з метою використання у різних предметних областях.

Висновки: 1. Виявлено, що успішне функціонування ПЗ багато в чому залежить від правильної організації процесу виконання робіт з визначення та аналізу вимог до нього. При цьому необхідні всі відповідні компетенції потенційні фахівці-аналітики отримують у процесі вирішення конкретних завдань у різних предметних областях як під час навчання, так і протягом виробничої діяльності.

2. З'ясовано нагальні проблеми, які стосуються процедури визначення вимог до ПЗ, а також виявлено шляхи їх уникнення та пом'якшення. Встановлено, що різні методології розроблення ПЗ можуть по-різному організовувати роботу аналітика щодо реалізації процедур визначення та аналізу вимог до ПЗ. Кожна з цих процедур є початковою, тобто вони повністю завершуються ще до початку етапу проектування архітектури ПЗ та етапу його розроблення. Водночас, в ітеративних методологіях розроблення ПЗ процедури визначення та аналізу вимог до нього в різному обсязі є на кожній ітерації.

3. Встановлено, що процедура аналізу вимог до ПЗ може бути тривалою та ресурсовитратною, що вимагає від аналітиків детального знання предметної області та застосування тонких психологічних навиків. З'ясовано, що багатьма науковцями і практиками виокремлено такі основні етапи процедури аналізу вимог до ПЗ: визначення зацікавлених сторін у розробленні ПЗ; інтерв'ю із зацікавленими сторонами щодо їхніх потреб від майбутнього ПЗ; спільні сесії визначення вимог до ПЗ; набори вимог до ПЗ у стилі контракту; вимірювані цілі застосування майбутнього ПЗ; прототипи – макети майбутнього ПЗ; прецеденти – технологія документування вимог до ПЗ; специфікація вимог до ПЗ.

4. Виявлено, що часто аналітики під час роботи з представниками від організації-замовника забувають чи не хочуть доводити до їхнього відома показники якості, яким має відповідати майбутнє ПЗ, знімаючи тим самим з себе певну відповідальність за виконану роботу. Згідно з міжнародними та вітчизняними стандартами оцінювання рівня якості програмного продукту, виділяють два процеси забезпечення якості ПЗ впродовж його ЖЦ: 1) *гарантія якості ПЗ*, що є результатом певних дій на кожній стадії його ЖЦ з перевірки й підтвердження відповідності його стандартам і процедурам, орієнтованим на досягнення певних характеристик якості; 2) *інженерія якості ПЗ* як процес надання майбутньому ПЗ надійності, супроводу й інших характеристик якості.

5. З'ясовано, що модель процесу визначення та аналізу вимог до ПЗ – це структура процесів його ЖЦ, що мають бути виконані від початку зародження проекту майбутнього ПЗ і доти, доки не будуть визначені та сформовані набори вимог, детально проаналізовані та погоджені сформульовані вимоги із зацікавленими сторонами. Також у цій моделі мають бути передбачені процедури введення відповідних змін до вимог, внесення цих змін у поточній версії ПЗ чи їх реструктуризація в наступних релізах. У формалізованій моделі удосконалено відповідні процедури, які стосуються визначення та аналізу вимог до ПЗ.

Перелік використаних джерел

Belchikov, Ya., & Birshtejn, M. M. (1989). *Delovye igry*. Riga: AVOTS, 304 p. [in Russian].

Blagodatskih, V. A., Volnin, V. A., & Poskakalov, K. F. (2005). *Standartizacija razrabotki programmnyh sredstv: uchebn. posob*. Moscow: Finansy i statistika, 288 p. [in Russian].

Bobalo, Yu. Ya., Volochii, B. Yu., Lozynskyi, O. Yu. et al. (2013). *Matematychni modeli ta metody analizu nadiinosti radioelektronnykh, elektrotekhnichnykh ta prohramnykh system: monohrafiia*. Lviv: Vyd-vo Lvivskoi politekhniky, 300 p. [in Ukrainian].

Boegh, J., Depanfilis, S., Kitchenham, B., & Pasquini, A. A. (1999). Method for Software Quality Planning, Control, and Evaluation. *IEEE Software, 16*(2), 69–77. https://doi.org/10.1109/52.754056

Futrell, R. T., Shafer, D. F., & Shafer, L. I. (2003). *Quality software project management*. New York: Prentice Hall PTR, 1136 p.

Gilb, T. (1988). *Principles of Software Engineering Management*. Reading MA: Addison Wesley, 464 p.

IEEE 1031-2011. (2008). IEEE Guide for the Functional Specification. Retrieved from: https://standards.ieee.org/findstds/standard/1031-2011.html

IEEE 1061-26157. (1998). Standard for Software Quality Metrics Methodology. Retrieved from: http://www.techstreet.com/cgi-bin/detail?product_id = 26157 (data obrashhenija: 20.06.2008).

IEEE 830-1998. (1998). *Recommended Practice for Software Requirements Specifications*. New York: IEEE, 44 p.

ISO/IEC, ISO/IEC 25000. (2005). Software Engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE. – Geneva: International Organization for Standardization. Retrieved from: http://www.iso.org/iso/iso_catalogue/ catalogue_tc/catalogue_detail.htm?csnumber = 35683 (data obrashhenija: 26.07.2008).

Kobern, A. (2011). *Sovremennye metody opisanija funkcionalnyh trebovanij k sistemam*. Moscow: Lori, 263 p. [in Russian].

Kungurcev, A. B., Kalinina, S. A., & Novikova, N. A. (2013). Model processa opredelenija trebovanij k programmnomu produktu. *Visnyk NTU "KhPI". Seriia: Novi rishennia v suchasnykh tekhnolohiiakh, 38*(1011), 55–61. Kharkiv : Vyd-vo NTU "KhPI". [in Russian].

Kungurcev, A., Blazhko, A., & Marulin, S. (2010). Metodika provedenija obuchenija osnovam proektirovanija programmnyh sistem v vide rolevyh kompjuternyh igr. Godishnik na tehnicheski universitet. In 3 vols. Vol. 1: Sbornik dokladov ot jubilejnoj nauchnoj konferencii s mezhdunarodnym uchastiem (pp. 123–126). Varna: Varna. [in Russian].

Lavrishheva, E. M., & Petruhin, V. A. (2006). *Metody i sredstva inzhenerii programmnogo obespechenija: uchebn. posob*. Moscow: Izd-vo Moskovskogo fiziko-tehnicheskogo instituta, 304 p. [in Russian].

Leffingujell, D., & Uidrig, Don. (2002). *Principy raboty s trebovanijami k programmnomu obespecheniju*. Moscow–Sankt-Petersburg: Izd. dom "Viljams", 450 p. [in Russian].

Lukina, M. (2003). *Tehnologija intervju: uchebn. posobie [dlja VUZov]*. Moscow: Aspekt Press, 480 p. [in Russian].

McConnell, S. (2013). *Sovershennyj kod. Master-klass*. Moscow: Izd-vo "Russkaja redakcija", 896 p. [in Russian].

McCall, J., Richards, P., & Walters, G. (1977). Factors in Software Quality. Three volumes: NTIS AD-A049-014, AD-A049-015, AD-A049-055. Retrieved from: http://oai.dtic.mil/oai/oai?&verb = getRecord&metadataPrefix = html&identifier = ADA049014 (data obrashhenija: 17.05.2008).

Mishhenko, V. O., Pomorova, O. V., & Govorushhenko, T. A. (2012). CASE-ocenka kriticheskih programmnyh sistem. In 3 vols. Vol. 1: Kachestvo; pod red. V. S. Kharchenko. Kharkov: NAU "KhAI", 201 p. [in Russian].

Pomorova, O. V., & Hovorushchenko, T. O. (2013). Suchasni problemy otsiniuvannia yakosti prohramnoho zabezpechennia. *Radioelektronni ta kompiuterni systemy, 5*, 319–327. Kharkiv : NAU "KhAI". [in Ukrainian].

Vedenina, V. (2013). *Delovaja igra i ee vozmozhnosti*. Retrieved from: http://tolerance.ru/teacher/kabinet/business-game.html (data obrashhenija: 26.02.2013). [in Russian].

Vigers, K. (2004). *Razrabotka trebovanij k programmnomu obespecheniju : per. s angl*. Moscow: Izd.-torg. dom "Russkaja Redakcija", 576 p. [in Russian].

Yakovyna, V., Seniv, M., Chabaniuk, Ya., Fedasiuk, D., & Khimka, U. (2010). Kryterii dostatnosti protsesu testuvannia prohramnoho zabezpechennia. *Visnyk Natsionalnoho universytetu "Lvivska politekhnika". Seria: Kompiuterni nauky ta informatsiini tekhnolohii, 672*, 346–358. [in Ukrainian].

Ю. И. Грыцюк, И. Ф. Лешкевич

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

ОСОБЕННОСТИ ОПРЕДЕЛЕНИЯ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ И ПРОБЛЕМЫ ИХ АНАЛИЗА

Проведен системный анализ предметной области, результаты которого позволили достоверно описать существующие проблемы, касающиеся процедур определения и анализа требований к программному обеспечению (ПО), выяснено виды деятельности аналитика для эффективного решения этих проблем, а также предложены основные показатели качества будущего ПО, усовершенствована формализованная модель для реализации процедур определения и анализа требований к ПО. Установлено, что процедура анализа требований к ПО может быть длительной и ресурсозатратной, что требует от аналитиков детального знания предметной области и применение тонких психологических навыков. Выяснено, что многими учеными и практиками выделены основные этапы процедуры анализа требований к ПО. При этом часто аналитики в процессе работы с представителями от организации-заказчика забывают или не хотят доводить до их сведения показатели качества, которым должно соответствовать будущее ПО, снимая тем самым с себя определенную ответственность за выполненную работу. Согласно международным и отечественным стандартам оценки уровня качества программного продукта, выделяют два процесса обеспечения качества ПО в течение его ЖЦ: гарантия качества ПО является результатом определенных действий на каждой стадии его ЖЦ по проверке и подтверждения соответствия его стандартам и процедурам, ориентированным на достижение определенных характеристик качества; инженерия качества ПО как процесс предоставления будущего ПО надежности, сопровождения и других характеристик качества.

Ключевые слова: предметная область; системный анализ предметной области; виды деятельности аналитика; определение требования к программному обеспечению (ПО); анализ требования к ПО; показатели качества ПО; модель определения и анализа требований к ПО.

Yu. I. Grytsiuk, I. F. Leshkevych

Lviv Polytechnic National University, Lviv, Ukraine

THE PROBLEMS OF DEFINITION AND ANALYSIS OF SOFTWARE REQUIREMENTS

Successful operation of the software largely depends on the proper organization of work to identify and analyze its requirements. Therefore the study aims at systematic analysing of the subject area. The results obtained enabled certain describing of the immediate problems related to procedures, and also identifying and analysing software requirements. The analysis has found activities to effectively address these issues and provide the main indicators of the quality of future software. A formalized model for implementation of procedures to identify and analyse software requirements is improved. The study has defined that potential experts can gain all the relevant competences in the process of solving specific problems in different subject areas both during training and production activities. The authors have highlighted the procedure for software requirements analysis can be quite extensive and resource intensive requiring detailed knowledge of the subject area and use subtle psychological skills. Moreover, many scientists and practitioners singled out the following main stages of the procedure of software requirements analysis: definition of interested parties in software development; interviews with interested parties on their needs in the future of software; joint session for definition of requirements for the software; sets of the requirements for the software contract in style; measurable targets for future software application; prototype – a model of future software; precedents – the technology for documenting requirements for software; specification of requirements for software. To conclude, firstly for proper software operation all the certain competences of field experts are required when performing specific tasks in various subject areas both during training and production activities. Secondly, urgent problems concerning procedures of determining requirements for software and the ways of their avoidance and mitigation are distinguished. Finally, appropriate procedures regarding the definition and analysis of software requirements are improved in the formalized model.

Keywords: systems analysis; subject area; analyst activities; definition of requirements for software; software requirements analysis; quality software; model definition and analysis of software requirements.

Інформація про авторів:

Грицюк Юрій Іванович, д-р техн. наук, професор, Національний університет "Львівська політехніка", м. Львів, Україна.
Email: yurii.i.hrytsiuk@lpnu.ua

Лешкевич Ігор Федорович, магістрант, Національний університет "Львівська політехніка", м. Львів, Україна.
Email: lif95.mailbox@gmail.com