УДК 004.421.2:004.05

УДОСКОНАЛЕНА МЕТОДИКА АНАЛІТИЧНОЇ КОНЦЕПТУАЛІЗАЦІЇ ПРЕДМЕТНОЇ ОБЛАСТІ НА ЕТАПІ ФОРМУЛЮВАННЯ ВИМОГ

Я. Дмитренко, група КН-24
А. Коваленко, асистент
Центральноукраїнський національний технічний університет

Успішне проєктування програмного забезпечення значною мірою залежить від якості аналізу предметної області та формулювання вимог. Аналіз предметної області — це перший і ключовий етап у життєвому циклі створення програмного забезпечення. Цей етап дає можливість кращого розуміння задачі, визначення користувачів, функцій і очікувань щодо майбутньої системи. На етапі аналізу ПрО критично важливо забезпечити точність та узгодженість функціональних та нефункціональних вимог. Удосконалення методики аналітичної концептуалізації дозволяє не лише покращити формування вимог, а й мінімізувати ризики проєктних помилок.

Метою роботи є удосконалення існуючих методик аналізу предметної області з метою підвищення точності формулювання функціональних і нефункціональних вимог до програмної системи.

Для досягнення високого рівня формалізації предметної області та чіткого формулювання вимог до програмної системи доцільно застосовувати візуальні засоби моделювання: діаграми потоків даних (DFD), уніфіковану мову моделювання (UML) та модель "сутність-зв’язок" (ER).

DFD (Data Flow Diagram) – застосовується для ілюстрації потоків даних між компонентами системи. Діаграми DFD дозволяють визначити, як інформація надходить до системи, як вона обробляється та зберігається.

UML (Unified Modeling Language) – забезпечує створення структурних і поведінкових моделей системи. Діаграми варіантів використання дозволяють виявити функціональні вимоги, а діаграми класів – структуру об'єктів системи.

ER-модель (Entity-Relationship) – дозволяє описати логічну структуру бази даних системи через сутності та зв’язки між ними. Це полегшує формулювання нефункціональних вимог, пов’язаних із зберіганням та доступом до інформації.

Запропоноване удосконалення методики полягає у комбінованому використанні DFD для опису потоків даних, UML для моделювання сценаріїв взаємодії користувача із системою, та ER-моделей для структурування інформаційної бази. Такий підхід дозволяє комплексно представити предметну область на процесному, поведінковому та інформаційному рівнях.

Правильно сформульовані вимоги грають основну роль у процесі розробки ПЗ для успішної реалізації будь-якого проєкту. Здебільшого вони поділяються на дві основні категорії: функціональні та нефункціональні. Вони виконують різні функції у формуванні кінцевої програмної системи. Функціональні вимоги визначають, які саме функції повинна виконувати система, акцентуючи увагу на її діях, поведінці та механізмах взаємодії з користувачем або іншими системами. Натомість нефункціональні вимоги описують якісні характеристики системи, такі як її продуктивність, безпечність, масштабність та інші критерії, що впливають на зручність та ефективність експлуатації.

Узгоджене розуміння відмінностей між цими видами вимог є необхідною складовою для забезпечення не лише функціональності системи відповідно до її призначення, а й високого рівня користувацького досвіду.

Функціональні вимоги – це конкретні технічні характеристики, що визначають, які операції має виконувати система, як вона обробляє вхідні дані, взаємодіє з користувачами, відповідає на події. Саме вони є основою для оцінки відповідності програмного продукту

поставленим цілям і очікуванням користувачів.

Нефункціональні вимоги, у свою чергу, зосереджені не на функціях, а на тому, як система виконує ці функції. Вони охоплюють загальні властивості – швидкодію, надійність, зручність користування, масштабованість, що забезпечують якість роботи та задоволення потреб користувачів. Їх дотримання гарантує відповідність програмного забезпечення сучасним стандартам та підвищує його конкурентоспроможність.

Нефункціональні вимоги, хоча часто розглядаються як другорядні, мають не менш важливе значення для успіху системи. Вони встановлюють обмеження і атрибути якості, які повинні бути дотримані незалежно від функціонального навантаження. Сюди належить здатність системи залишатися працездатною у випадку часткових збоїв, ефективно обробляти зростання обсягу користувачів та захищати дані від несанкціонованого доступу. Вимоги до масштабованості можуть передбачати як вертикальне, так і горизонтальне розширення ресурсів системи без зниження продуктивності.

Інтеграція нефункціональних вимог у процес розробки має важливе значення для запобігання значним витратам на етапі тестування та підтримки. Якщо такі вимоги визначені недостатньо чітко або ігноруються, зростає ймовірність виникнення дефектів, що впливають на стабільність, продуктивність або безпеку системи.

Варто також зазначити, що в процесі еволюції системи нефункціональні вимоги можуть змінюватися частіше, ніж функціональні. Зміни в нормативно-правових актах, технологічних стандартах або очікуваннях користувачів можуть вимагати оновлення вимог до безпеки, доступності чи продуктивності. Тому важливо впроваджувати підходи гнучкого управління вимогами, які дозволяють швидко адаптувати проєктні рішення відповідно до нових умов.

Функціональні та нефункціональні вимоги є критично важливими для всіх етапів життєвого циклу програмного забезпечення, від початкового проєктування до тестування і подальшої підтримки. Функціональні вимоги зазвичай оформлюються у вигляді специфікацій сценаріїв використання та схем обробки даних, що описують взаємодію користувача із системою та визначають умови виконання необхідних операцій. Вони забезпечують чітке розуміння того, які завдання повинні виконуватися, наприклад, авторизація користувачів, обробка транзакцій або формування звітності.

Висновки. Без чіткого аналізу предметної області неможливо сформувати обґрунтоване технічне завдання або бачення системи. Це призводить до втрати часу, ресурсів і можливих помилок під час реалізації. Тому якісний аналіз предметної області дає можливість уникнути невизначеності, зменшити ризики, підвищити якість програмного продукту та забезпечити його відповідність реальним потребам користувачів. Аналіз дозволяє визначити, що саме система повинна робити, та якою вона має бути (швидкодія, зручність, безпека тощо). Успішне керування як функціональними, так і нефункціональними вимогами має вирішальне значення для розробки програмного забезпечення, яке відповідає потребам користувачів, забезпечуючи при цьому оптимальну продуктивність, надійність і задоволення користувачів.

Список літератури

1. ISO/IEC/IEEE 29148:2018. Systems and software engineering — Life cycle processes — Requirements engineering. URL: https://www.iso.org/standard/72089.html .
2. Sommerville, I. Software Engineering. 10th ed. – Boston: Pearson Education, 2016. – 792 p.
3. Visure Solutions. Функціональні та нефункціональні вимоги (з прикладами) [Електронний ресурс]. URL: https://visuresolutions.com/uk/blog/funktsionalni-ta-nefunktsionalni-vymogy-z-prykladamy.
4. QATestLab. Нефункціональні вимоги: приклади, типи, підходи [Електронний ресурс] // QATestLab. – 2023. URL: https://training.qatestlab.com/blog/technical-articles/non-functional-requirements-examples-types-approaches.

Войчур Ю., Медзатий Д. Метод аналізу вимог до програмного забезпечення на предмет пошуку значень атрибутів якості // Measuring and Computing Devices in Technological Processes. – 2024. – №1. – С. 146–151. – DOI: 10.31891/2219-9365-2024-77-18