УДК 004.415.2:004.8

ВИЯВЛЕННЯ НЕОДНОЗНАЧНОСТІ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

Попова М.О., к.е.н., доц.
Національний університет «Одеська політехніка»
Україна, Одеса
popova.m.o@op.edu.ua

Анотація. *Запропоновано процедуру виявлення неоднозначності програмних вимог шляхом побудови й зіставлення альтернативних тлумачень із використанням великих мовних моделей. Визначено структуру результату аналізу та порядок його експертного підтвердження. На сконструйованому прикладі показано вплив проєктного контексту на оцінювання вимоги. Процедура призначена для підтримки перевірки специфікацій.*

Ключові слова: *інженерія вимог; неоднозначність вимог; великі мовні моделі; якість специфікації; контекстний аналіз.*

Вступ. Вимоги, викладені природною мовою, забезпечують обмін інформацією між замовником, аналітиком і командою розроблення, проте допускають різне розуміння очікуваної поведінки системи. Неоднозначність виникає, коли одне формулювання має щонайменше два обґрунтовані тлумачення. Її наслідки виявляються під час проєктування, реалізації та приймання програмного забезпечення: учасники погоджують текст вимоги, але вкладають у нього різний зміст.

Автоматизована перевірка мовних конструкцій допомагає знаходити потенційно проблемні фрагменти. Зокрема, у дослідженні F. Talha та співавторів розглянуто виявлення неоднозначних займенникових посилань, сполучникових конструкцій і пропущених умов із використанням методів оброблення природної мови [1]. Водночас сама наявність певного слова чи синтаксичної конструкції ще не доводить неоднозначності конкретної вимоги. Для такого висновку необхідно враховувати термінологію та правила предметної області.

Великі мовні моделі відкривають можливість аналізувати не лише окремі мовні ознаки, а й альтернативні значення висловлювання. Дослідження G. Kamath та співавторів показує здатність низки моделей розрізняти тлумачення речень із неоднозначною областю дії семантичних операторів [2]. Однак ці результати не можна безпосередньо переносити на перевірку програмних специфікацій. У роботі V. V. Poulsen та співавторів продемонстровано значення предметного контексту для виявлення неоднозначності вимог [3]. Отже, застосування мовної моделі потребує процедури, яка пов’язує її висновок із текстом вимоги, доступним контекстом і рішенням аналітика.

Метою дослідження є підвищення якості перевірки вимог до програмного забезпечення щодо їхньої однозначності шляхом застосування великих мовних моделей для побудови альтернативних тлумачень, перевірки їхньої сумісності з проєктним контекстом та подальшого експертного підтвердження виявленої неоднозначності.

Основна частина. У дослідженні використано аналіз наукових публікацій, моделювання процедури перевірки та порівняння можливих тлумачень вимоги на сконструйованому прикладі. Об’єктом аналізу є окрема вимога разом із фрагментами специфікації, необхідними для її розуміння. До контексту належать визначення термінів,

ролі користувачів, бізнес-правила та пов’язані вимоги. Аналіз ізольованого речення розглядається лише як попередній етап.

Передусім розмежовано неоднозначність, нечіткість і неповноту. Формулювання «система повинна швидко відкривати звіт» містить невизначений критерій швидкодії. Його недолік полягає у відсутності вимірюваної межі прийнятності. Натомість займенник, який може стосуватися двох різних учасників процесу, створює альтернативні тлумачення того, хто має виконати дію або отримати результат. Пропущена умова також може спричинити різне розуміння, але спочатку потребує перевірки повноти опису. Це розмежування потрібне, щоб модель не позначала будь-яку проблему якості загальною категорією неоднозначності.

Запропонована процедура починається з підготовки пакета вхідних даних: ідентифікатора й редакції вимоги, її повного тексту та відібраного контексту. Для кожного контекстного фрагмента зберігається посилання на джерело. Інформація, яку модель може припустити на основі загальних знань, не прирівнюється до погодженого правила проєкту.

На наступному етапі модель визначає фрагмент, який допускає різне розуміння, і формулює альтернативні тлумачення. В інструкції до аналізу встановлюється вимога зберігати зміст початкового формулювання та не додавати нових функцій системи. Кожне тлумачення має супроводжуватися поясненням, на яку мовну конструкцію воно спирається. Якщо для побудови альтернативи доводиться вводити відсутнє бізнес-правило, така альтернатива позначається як припущення.

Далі перевіряється, чи спричиняють тлумачення різну спостережувану поведінку системи. Для цього визначаються учасник, дія, об’єкт, умова виконання та очікуваний результат. Наприклад, відмінності в одержувачі повідомлення або межах доступу можуть змінити реалізацію та критерії приймання. Звичайне перефразування, яке зберігає однакову поведінку, не є достатнім свідченням неоднозначності.

Після зіставлення з контекстом формується один із трьох результатів: «потенційна неоднозначність», «ознак неоднозначності не виявлено» або «недостатньо контексту для висновку». Перший результат передбачає наявність щонайменше двох обґрунтованих тлумачень із різними поведінковими наслідками, які не виключені наданими правилами. Третій застосовується, коли без додаткових відомостей неможливо оцінити допустимість запропонованих тлумачень. Відсутність виявлених ознак не означає доведеної однозначності вимоги.

Результат оформлюється як картка перевірки. Вона містить ідентифікатор і редакцію вимоги, проблемний фрагмент, альтернативні тлумачення, відмінності очікуваної поведінки, посилання на використаний контекст та запитання до зацікавленої сторони. Окремо фіксується рішення аналітика: підтвердити зауваження, відхилити його з поясненням або запросити уточнення. Така структура дає змогу перевірити підстави висновку й повернутися до них після зміни специфікації.

Розглянемо сконструйовану вимогу: «Менеджер передає заявку адміністратору, після чого система надсилає йому повідомлення». Займенник «йому» допускає віднесення до менеджера або адміністратора. У першому випадку система повідомляє менеджера про передавання заявки, у другому – адміністратора про її надходження. Тлумачення передбачають різних одержувачів і, відповідно, різні перевірки під час приймання.

Якщо проєктний контекст не визначає одержувача, підставою для зауваження є обидва тлумачення, а запитання до замовника формулюється без вибору одного з них: «Хто повинен отримати повідомлення після передавання заявки?». Якщо ж погоджене бізнес-правило прямо встановлює одержувачем адміністратора, контекст знімає невизначеність поведінки. Водночас аналітик може рекомендувати замінити займенник назвою ролі, щоб вимога залишалася зрозумілою під час її окремого використання.

Уточнена редакція після підтвердження замовником може мати вигляд: «Після передавання заявки менеджером система надсилає повідомлення адміністратору, якому передано заявку». Це уточнення не встановлює каналу повідомлення, строку доставки чи його змісту: такі характеристики потребують окремого погодження. Таким чином, усунення неоднозначності не повинно перетворюватися на непомітне розширення вимоги.

Для відтворюваності аналізу необхідно зберігати назву та версію моделі, текст інструкції, налаштування генерації й склад контекстного пакета. Зміна будь-якого з цих елементів може впливати на результат. Запропонована процедура не залежить від конкретного постачальника моделі, однак порівняння моделей потребує однакових вхідних матеріалів і критеріїв оцінювання.

Подальшу експериментальну перевірку доцільно виконувати на наборі вимог із незалежним експертним розміченням і погодженням спірних випадків. Слід порівняти аналіз без контексту, аналіз із контекстом та просту перевірку мовних ознак. Оцінюванню підлягають точність і повнота виявлення, обґрунтованість альтернативних тлумачень та кількість запропонованих змін, які додають непогоджені функції. Для використання в навчанні додатково можна оцінювати здатність студента пояснити рішення й обґрунтувати прийняття або відхилення зауваження моделі.

Висновки. Запропоновано процедуру виявлення потенційної неоднозначності програмних вимог, у якій висновок мовної моделі обґрунтовується альтернативними тлумаченнями, відмінностями очікуваної поведінки та проєктним контекстом. Визначено структуру картки перевірки й порядок експертного підтвердження. На сконструйованому прикладі показано, що одне формулювання може потребувати уточнення за відсутності бізнес-правила та мати визначене тлумачення за його наявності. Практичне застосування процедури полягає в підтримці рецензування специфікацій і навчанні аргументованого аналізу вимог. Її результативність потребує подальшого експериментального оцінювання на розміченому наборі вимог.

Використання інструментів ШІ. ChatGPT (OpenAI, модель GPT-6) використано для мовного редагування анотацій, вступу, основної частини й висновків. Експериментальне оцінювання мовних моделей у межах цієї роботи не проводилося.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Talha, F., Tahir, T., & Nadeem, T. (2025). A semiautomated approach for detecting ambiguities in software requirements using SpanBERT and named entity recognition. *Journal of Software: Evolution and Process, 37*(8), e70041. https://doi.org/10.1002/smr.70041

[2] Kamath, G., Schuster, S., Vajjala, S., & Reddy, S. (2024). Scope ambiguities in large language models. *Transactions of the Association for Computational Linguistics, 12*, 738-754. https://doi.org/10.1162/tacl_a_00670

[3] Poulsen, V. V., Guertler, M., Eisenbart, B., Tomidei, L., & Sick, N. (2026). Context-aware large language models for ambiguity detection in requirements. *Proceedings of the Design Society, 6*, 2493-2502. https://doi.org/10.1017/pds.2026.10607

Popova M.O.

Detecting Ambiguity in Software Requirements Using Large Language Models

Abstract*. A procedure for detecting ambiguity in software requirements through the generation and comparison of alternative interpretations using large language models is proposed. The analysis record and expert review process are defined. A constructed example illustrates how project context affects requirement assessment. The procedure supports specification reviews.*

Keywords*: requirements engineering; requirements ambiguity; large language models; specification quality; contextual analysis.*