УДК 004.41:174:004.8

ПРОФЕСІЙНА ВІДПОВІДАЛЬНІСТЬ ПРОГРАМНОГО ІНЖЕНЕРА ПІД ЧАС ВИКОРИСТАННЯ ГЕНЕРАТИВНОГО ШІ

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

Анотація. *Обґрунтовано зв'язок професійної відповідальності програмного інженера з перевіркою результатів генеративного ШІ. Запропоновано процедуру їх прийняття з фіксацією доказів і розподілом обов'язків у команді. На аналітичному прикладі показано умови відхилення неперевіреного рішення. Результати придатні для організації рецензування та професійної підготовки інженерів.*
Ключові слова: *програмна інженерія; професійна відповідальність; генеративний ШІ; інженерна етика; перевірка результатів.*

Вступ. Генеративний штучний інтелект (ШІ) використовується для підготовки програмного коду, тестів, вимог і технічної документації. Отриманий матеріал може виглядати переконливо, навіть коли містить помилки або спирається на неперевірені припущення. Тому професійного значення набуває рішення інженера про включення такого результату до програмного продукту. Посилання на автоматичне походження фрагмента не пояснює, чому його визнано придатним до використання.

Етичний кодекс Асоціації обчислювальної техніки (ACM) пов'язує професійну діяльність із якістю роботи, компетентністю, рецензуванням і всебічним оцінюванням ризиків [1]. Ці принципи створюють основу для аналізу використання генеративного ШІ, але потребують конкретизації в щоденних процедурах команди. Зокрема, необхідно встановити, хто перевіряє результат, які докази достатні для його прийняття та як діяти за відсутності необхідної компетентності.

Емпіричне дослідження N. Perry та співавторів виявило, що учасники, які користувалися асистентом на основі codex-davinci-002, у досліджених завданнях створювали менш безпечний код і водночас частіше вважали його безпечним [2]. Цей висновок стосується конкретного експерименту й не є оцінкою всіх сучасних інструментів. Проте він засвідчує потребу розмежовувати впевненість користувача та результати перевірки. Профіль ризиків генеративного ШІ Національного інституту стандартів і технологій США (NIST) розглядає помилкові результати, взаємодію людини з ШІ та організацію нагляду [3]. Практичним питанням залишається перехід від загальних рекомендацій до обґрунтованого рішення щодо конкретної зміни.

Метою роботи є зниження ризику використання неперевірених результатів генеративного ШІ та підвищення обґрунтованості інженерних рішень шляхом установлення процедур перевірки, документування підстав прийняття результатів і розподілу обов'язків між учасниками команди розроблення.

Основна частина. У роботі виконано зіставлення професійно-етичних принципів із ризиками використання генеративного ШІ та аналітичне моделювання процедури прийняття його результатів. Професійну відповідальність розглянуто як обов'язок фахівця обґрунтовувати рішення в межах своєї компетентності, повідомляти про суттєві обмеження й

забезпечувати належну перевірку роботи. Таке розуміння охоплює дії до виникнення дефекту, а також реагування на вже виявлену проблему.

Результатом застосування ШІ може бути окремий фрагмент коду, тест, вимога або пояснення. Одиницею оцінювання доцільно обирати зміну, яку команда планує прийняти, а не відповідь моделі загалом. Для неї встановлюються призначення, пов'язані вимоги, можливі наслідки помилки та умови використання. Код, що формує допоміжне повідомлення, і код перевірки доступу потребують різного обсягу перевірки, оскільки помилки мають різні наслідки.

Запропонована процедура починається до передавання даних інструменту. Інженер визначає допустимість використання обраного сервісу для конкретного завдання та мінімально необхідний склад вхідних матеріалів. Доступність даних у робочому середовищі не означає дозволу передавати їх зовнішньому сервісу. Для прикладів і перевірок використовуються синтетичні або належно підготовлені дані; особливості зберігання запитів і доступу до них враховуються відповідно до погоджених правил організації.

Після отримання результату фахівець з'ясовує його логіку, припущення та залежності. Для коду перевіряються відповідність вимогам, оброблення помилок і вплив на пов'язані компоненти; для документації – достовірність тверджень і посилань; для вимог – походження обмежень та їх погодження. Виявлені зовнішні компоненти й запозичені матеріали перевіряються щодо джерел та умов використання. Невідоме походження не повинно приховуватися під позначкою про генерацію.

Далі визначається незалежна основа перевірки. Тести мають випливати з погоджених вимог і ризиків, а не лише повторювати припущення згенерованого коду. Звернення до тієї самої моделі з проханням підтвердити правильність відповіді не замінює виконання тестів або фахового рецензування. Автоматично сформовані тести також потребують перевірки очікуваних результатів. Для змін, пов'язаних із доступом до даних чи іншими суттєвими ризиками, залучається рецензент із відповідною компетентністю.

Рішення про використання результату фіксується разом із доказами: пов'язаними вимогами, виконаними перевірками, висновком рецензента та невирішеними обмеженнями. Достатність доказів оцінюється щодо призначення зміни. Коментар «перевірено» без зазначення способу перевірки не дозволяє відновити підстави рішення. За недостатності відомостей результат повертається на доопрацювання або передається фахівцю; суттєва невирішена невизначеність є підставою відкласти його прийняття.

Розподіл обов'язків має відповідати повноваженням учасників. Автор зміни пояснює її поведінку й надає результати перевірки; рецензент оцінює визначені аспекти в межах своєї компетентності; уповноважений учасник ухвалює рішення про включення зміни до продукту. Керівник забезпечує час, доступ до фахівців і зрозумілі правила використання інструментів. Один учасник може поєднувати ролі, але це поєднання повинно бути явним. Сам факт участі рецензента не скасовує обов'язків автора, а відповідальність організації за процес не зводиться до підпису виконавця.

Розглянемо сконструйований приклад: інженер за допомогою генеративного ШІ підготував функцію експорту замовлень користувача. Функція приймає ідентифікатор користувача із запиту й повертає відповідні записи, але не перевіряє, чи має поточний користувач право на їх перегляд. Тест із власним ідентифікатором завершується успішно. Водночас вимога про ізоляцію даних між обліковими записами залишається неперевіреною.

У запропонованій процедурі підставою оцінювання стає погоджене правило доступу: користувач може експортувати лише власні замовлення, якщо інші повноваження не надано

окремо. Перевірка звернення з ідентифікатором іншого користувача виявляє відсутність необхідного обмеження. Зміну повертають на доопрацювання, після чого перевіряють дозволений доступ, відмову в недозволеному доступі та відсутність чужих записів у відповіді. Конкретний механізм авторизації обирається відповідно до архітектури системи. Це аналітичний приклад, а не звіт про проведений програмний експеримент.

Приклад показує, що працездатність функції в одному сценарії не є достатньою підставою для її професійного схвалення. Потрібні зв'язок із вимогою та перевірка наслідків помилки. Для збереження підстав рішення запропоновано мінімальний запис: призначення зміни, інструмент і доступний ідентифікатор моделі, використані категорії даних, суттєві припущення, результати перевірок, рецензент і рішення про прийняття. Повні запити з конфіденційними матеріалами не слід автоматично дублювати в такому записі.

Після прийняття зміни відповідальність продовжується через оброблення повідомлень про дефекти й перегляд попередніх рішень. У професійній підготовці цю процедуру можна використовувати як основу розбору випадків: студент пояснює запропонований фрагмент, визначає ризик і надає аргументи за його прийняття або відхилення. Предметом оцінювання стає обґрунтованість рішення, а не лише наявність працездатного результату.

Висновки. Конкретизовано професійну відповідальність програмного інженера під час використання генеративного ШІ через перевірку припущень, оцінювання ризиків і документоване прийняття результату. Запропоновано процедуру, яка пов'язує допустимість вхідних даних, незалежну основу перевірки та повноваження учасників команди. На аналітичному прикладі показано підставу відхилення функції, працездатної в основному сценарії, але такої, що не забезпечує погодженого обмеження доступу. Визначено мінімальний склад запису про прийняття зміни. Практичне застосування можливе в рецензуванні та навчанні професійної практики; вплив процедури на частоту дефектів потребує емпіричного оцінювання.

Використання інструментів ШІ. ChatGPT (OpenAI, модель GPT-6) використано для мовного редагування анотацій, вступу, основної частини й висновків. Емпіричні дані за допомогою ШІ не генерувалися.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Association for Computing Machinery. (2018). *ACM code of ethics and professional conduct*. https://www.acm.org/code-of-ethics

[2] Perry, N., Srivastava, M., Kumar, D., & Boneh, D. (2023). Do users write more insecure code with AI assistants? In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security* (pp. 2785-2799). Association for Computing Machinery. https://doi.org/10.1145/3576915.3623157

[3] Autio, C., Schwartz, R., Dunietz, J., Jain, S., Stanley, M., Tabassi, E., Hall, P., & Roberts, K. (2024). *Artificial intelligence risk management framework: Generative artificial intelligence profile* (NIST AI 600-1). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.AI.600-1

Popova M.O.

Professional responsibility of software engineers when using generative AI

Abstract. *Software engineers' professional responsibility is linked to the verification of generative AI outputs. An acceptance procedure with documented evidence and assigned team duties is proposed. A constructed example illustrates grounds for rejecting an unverified solution. The findings support professional review and software engineering education.*

Keywords: *software engineering; professional responsibility; generative AI; engineering ethics; output verification.*