УДК

Шишкін М.С.
Харківський національний економічний університет імені Семена Кузнеця, Харків, Україна

АГЕНТНИЙ ШІ В ЖИТТЄВОМУ ЦИКЛІ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ: ФАЗА РОЗРОБКИ

Вступ.

Фаза розробки - життєвого циклу розробки програмного забезпечення (Software development lifecycle, SDLC) традиційно зосереджена на ручному кодуванні, експертних оцінках та ітеративному налагодженні, що виконується розробниками-людьми за підтримки статичних інструментів. Поява агентного штучного інтелекту привела до зміни парадигми в цьому процесі, дозволивши системам штучного інтелекту діяти автономно, досягати цілей, координувати підзадачі та адаптувати рішення на основі зворотного зв'язку з навколишнього середовища. На відміну від традиційних інструментів автодоповнення коду, які працюють як пасивні помічники, агентні системи штучного інтелекту характеризуються орієнтацією на цілі, контекстним мисленням, делегуванням завдань та здатністю самостійно виконувати багатоетапні робочі процеси. Це перетворює фазу розробки з виключно людської діяльності на середовище для співпраці, де розробники-люди контролюють цілі високого рівня, тоді як агенти штучного інтелекту обробляють завдання рівня виконання.

Застосування агентного штучного інтелекту у фазі розробки.

У щоденній діяльності з розробки програмного забезпечення агентні системи штучного інтелекту працюють як автономні співробітники, здатні перетворювати абстрактні вимоги на робочі компоненти програмного забезпечення. Після отримання історій користувачів або технічних специфікацій ці агенти виконують семантичний аналіз для інтерпретації бізнес-намірів, визначення функціональних меж та розкладання високорівневих цілей на структуровані завдання розробки. На основі цього аналізу агенти генерують початкові скелети коду, визначають інтерфейси модулів та створюють заглушки сервісів, які відповідають вибраним архітектурним шаблонам, таким як мікросервіси або багаторівневі дизайни додатків. Контракти API визначаються програмно, що забезпечує узгодженість між визначеннями сервісів, моделями запитів та схемами відповідей, тоді як залежності від сторонніх розробників вибираються та інтегруються відповідно до міркувань сумісності та безпеки.

Агентний ШІ може орієнтуватися та розуміти складні існуючі кодові бази, щоб забезпечити відповідність нових змін встановленим стандартам розробки та конвенціям проекту. Агенти аналізують історію репозиторію, рекомендації щодо кодування та архітектурну документацію, щоб відповідати правилам іменування, стандартам форматування та межам модулів. Під час впровадження нових функцій автономні агенти здатні знаходити відповідні точки розширення в кодовій базі, впроваджувати логіку без порушення принципів інкапсуляції та виконувати локалізований рефакторинг, коли виявляється надвисока складність коду або шаблони дублювання. Цей безперервний процес рефакторингу покращує зручність обслуговування та підтримує зменшення технічного боргу одночасно з розгортанням функцій, а не відкладає такі зусилля на фази після релізу [1].

Автоматизоване тестування та інтеграція безпеки.

Робочий процес тестування також тісно інтегрується з агентною підтримкою розробки. Агенти ШІ динамічно генерують модульні та інтеграційні тести, узгоджені з новим кодом, і можуть автоматично виконувати ці тести в конвеєрах розробки. Коли з'являються збої тестування або попередження статичного аналізу, агенти інтерпретують журнали, виявляють потенційні першопричини та ітеративно змінюють код, доки не будуть виконані критерії стабільності. Паралельно агенти, орієнтовані на безпеку, сканують вихідні файли на наявність вразливостей, таких як недоліки ін'єкцій, небезпечні залежності або неправильні конфігурації,

та пропонують безпечні шаблони кодування, перш ніж недоліки поширяться в нижчі середовища. Цей цикл зворотного зв'язку в режимі реального часу забезпечує «безпеку зсуву вліво», де структурні проблеми та проблеми відповідності вирішуються безпосередньо на етапі розробки, а не після випуску.

Вплив на продуктивність та співпрацю.

Співпраця агентів забезпечує вимірне підвищення продуктивності. Розробники витрачають менше часу на повторювані завдання кодування та генерацію шаблонів і більше часу на моделювання рішень, проектування бізнес-логіки та перевірку архітектурних рішень. Цикли розробки функцій скорочуються, оскільки агенти ШІ працюють безперервно без перерв, а якість коду покращується завдяки вбудованому забезпеченню узгодженості та автоматизації тестування. Розробники-люди все частіше виконують роль супервайзерів, які керують стратегічними намірами, перевіряють результати та вирішують складні граничні випадки, що вимагають обґрунтування предметної області, що виходить за межі машинної інтерпретації.

Ризики та виклики управління.

Незважаючи на суттєві переваги в продуктивності та якості, що з'являються завдяки агентному ШІ на етапі розробки SDLC, його впровадження також створює значні технічні, організаційні та етичні проблеми, які вимагають ретельного управління. Одним з центральних ризиків є тенденція до надмірного делегування, коли команди розробників все більше покладаються на автономних агентів для прийняття рішень щодо проектування та логіки впровадження. Хоча таке делегування прискорює розробку, воно може поступово зменшувати безпосередню взаємодію розробників із семантикою програмного забезпечення, архітектурним мисленням та низькорівневими практиками налагодження. З часом ця залежність може підірвати важливі інженерні компетенції, роблячи команди менш здатними критично перевіряти рішення, створені ШІ, або ефективно реагувати на складні збої, які перевищують можливості автоматизованого мислення.

Складність управління зростає, оскільки відповідальність за розробку розподіляється між людьми та автономними агентами. Традиційні моделі відповідальності припускають простежуване людське авторство для інженерних рішень, тоді як агентна розробка впроваджує частково непрозорі ланцюжки рішень, керовані оперативною інженерією, внутрішніми міркуваннями та ймовірнісними механізмами генерації. Визначення відповідальності за дефекти, вразливості безпеки або невідповідність нормативним вимогам стає складнішим, особливо в середовищах, що підлягають суворим вимогам до аудиту або сертифікації. Ця проблема вимагає впровадження систем управління, які забезпечують простежуваність коду, згенерованого штучним інтелектом, збереження підказок генерації, відстеження версій моделі та документовані контрольні точки перевірки для забезпечення перевіреного нагляду. Ці обмеження підкреслюють необхідність структурованих процесів перевірки та формальних робочих процесів перевірки для підтримки підзвітності та довіри[2].

Висновок.

На завершення, агентний ШІ фундаментально змінює фазу розробки SDLC, переміщуючи робочі процеси, що вимагають виконання, до автономних систем, одночасно піднімаючи розробників-людей до супервайзерів та архітектурних ролей. Найефективніша модель впровадження поєднує агентну автономію з людським управлінням, гарантуючи, що підвищення продуктивності збалансовано забезпеченням якості, гарантіями безпеки та етичною відповідальністю. У міру розвитку агентних фреймворків та їхньої глибшої інтеграції в середовища розробки очікується, що вони пришвидшать розробку, підвищать надійність коду та переосмислять практики спільної розробки програмного забезпечення в хмарних та корпоративних екосистемах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. – 4-те вид. – Hoboken : Pearson, 2021. – 1152 с..
2. NIST. Artificial Intelligence Risk Management Framework (AI RMF 1.0). – Gaithersburg : National Institute of Standards and Technology, 2023. – 128 с