УДК 004.415

Б. Ю. Панасюк
Н.П. Бабюк
Ю.В. Сторожук

LLM-АСИСТОВАНИЙ РЕФАКТОРИНГ API-КОНТРАКТІВ: ТИПОВІ ПОМИЛКИ, GUARDRAILS, ОЦІНЮВАННЯ ЯКОСТІ

Вінницький національний технічний університет

Анотація

Метою роботи є розробка процедури LLM-асистованого рефакторингу API-контрактів OpenAPI з контролем якості через "guardrails". Розглянуто типові помилки, що виникають під час ручного оновлення специфікацій (неконсистентні параметри та схеми, невалідні фрагменти специфікації, пропуски відповідей і описів помилок, поява прихованих breaking changes). Запропоновано конвеєр рефакторингу: LLM генерує структурований план змін і патч до контракту, після чого застосовуються формальні перевірки валідності, лінтування стилю та автоматичне виявлення несумісних змін за diff-звітом. Визначено метрики якості та наведено ілюстративний псевдоексперимент на прикладі мікросервісного API. Отримані результати можуть бути використані в CI/CD для надійної еволюції контрактів у підході API-first.

Ключові слова: API-first, OpenAPI, LLM, рефакторинг контрактів, guardrails, сумісність API, продуктивність, оцінювання якості.

Abstract

The aim of this work is to develop a procedure for LLM-assisted refactoring of OpenAPI API contracts with quality control via "guardrails." Typical issues that arise during manual specification updates are examined, including inconsistent parameters and schemas, invalid specification fragments, missing responses and error descriptions, and the emergence of hidden breaking changes. A refactoring pipeline is proposed: an LLM generates a structured change plan and a contract patch, after which formal validation checks, style linting, and automated detection of incompatible changes using a diff report are applied. Quality metrics are defined, and an illustrative pseudo-experiment is presented using a microservice API as an example. The obtained results can be used in CI/CD to support reliable contract evolution within an API-first approach.

Keywords: API-first, OpenAPI, LLM, contract refactoring, guardrails, API compatibility, performance, quality assessment.

Вступ

У підході API-first контракт є "джерелом істини" для команд, клієнтів і автоматизації. На практиці OpenAPI-специфікації постійно еволюціонують: додаються поля та статус-коди, уточнюються схеми, уніфікуються помилки, вводяться правила іменування, безпека, версіонування. Рефакторинг контракту часто виконується вручну, що призводить до помилок двох типів: (1) формальних — контракт перестає бути валідним або втрачає узгодженість параметрів/схем; (2) еволюційних — у контракт тихо потрапляють несумісні зміни, які ламають існуючих споживачів. Використання мовних моделей великого розміру (LLM) потенційно скорочує час редагування, але підвищує ризик "галюцинацій" (вигадані поля/ендпоїнти), неявної зміни семантики та некерованих "покращень", які перетворюються на breaking changes.

Результати дослідження

OpenAPI Specification визначає стандартний, мовонезалежний опис HTTP API, який придатний як для людей, так і для інструментів автоматизації [1]. Випуск OpenAPI 3.1.0 декларує узгодження з JSON Schema (draft 2020-12), підсилюючи можливості формальної перевірки схем та уніфікацію семантики опису даних [2], [3]. Для зменшення помилок генерації в LLM-процесах застосовують структуровані виходи за JSON Schema та інструментальні виклики (tool/function calling), які дозволяють вимагати від

моделі строго визначений формат результату та будувати керовані конвеєри обробки [4], [5]. Для контролю якості API-контрактів широко використовуються лінтери (наприклад, Spectral із правилами для OpenAPI) та автоматичне виявлення несумісних змін через порівняння специфікацій [6], [7]. Дослідження LLM для задач програмування (зокрема Codex) підтверджують ефективність моделей у генерації/редагуванні артефактів розробки, але підкреслюють необхідність автоматизованої перевірки результатів [8]. Водночас методика LLM-асистованого рефакторингу саме контрактів OpenAPI з відтворюваними guardrails та метриками якості залишається практично важливою і недостатньо формалізованою.

Запропонована процедура (конвеєр) орієнтована на рефакторинг без зміни бізнес-семантики API, якщо інше не задано вимогою. Вхідні дані: (a) поточний контракт OpenAPI (YAML/JSON), (b) "запит на зміну" (change request), (c) правила стилю/політики (наприклад: назви, помилки, security), (d) попередня версія контракту (для compatibility-check). Вихід: патч до контракту + звіт перевірок.

Логіка конвеєра LLM-рефакторингу:

Запит на зміну → LLM: план змін + патч → Валідація OpenAPI/Schema → Lint (style) → Diff/Breaking-check → PR + рев'ю.

У конвеєрі LLM використовується як генератор двох артефактів:

1) План змін (ChangePlan) — короткий список кроків, яких торкається рефакторинг (які paths/operations/schemas зачіпаються і чому).

2) Патч (ContractPatch) — машинозчитуване представлення змін (наприклад, JSON Patch або перелік "операцій редагування"), що зменшує ризик випадкових правок поза вимогою. Структурованість і валідація виходу забезпечуються через JSON Schema [4], а інструментальні виклики — для запуску перевірок у пайплайні [5].

Типові помилки, які доцільно контролювати автоматично, згруповано так:

(A) Синтаксичні/формальні: невалідний YAML/JSON, порушення структури OpenAPI, некоректні $ref.

(B) Узгодженість інтерфейсу: параметр у path відсутній у параметрах операції; невідповідність required/nullable; розрив між requestBody та content-type.

(C) Якість опису/документації: відсутні summary/description, теги, приклади, зрозумілі назви схем.

(D) Помилки доменної семантики: модель змінює зміст поля (тип/одиниці/enum) без явного запиту.

(E) Breaking changes: видалення/перейменування endpoint; посилення вимог (новий required-параметр); звуження enum; зміна типу поля; зміна відповіді без підтримки старого варіанта.

Пропонований guardrails-чекліст (мінімальний, для CI):

1) Заборонити "вільне редагування": LLM має повертати лише патч у заздалегідь визначеному форматі (структурований вихід) [4].

2) Валідація специфікації: контракт після патча має проходити формальну перевірку OpenAPI + коректність схем даних, узгоджених із JSON Schema 2020-12 [1], [3].

3) Лінтування стилю: контракт має задовольняти організаційні правила (наприклад, naming, security, errors) через Spectral ruleset [6].

4) Breaking-check: для "не-ревізійних" змін патч не має вводити breaking changes; автоматичне виявлення — через OpenAPI diff (breaking changes report) [7].

5) Обмеження області: дозволяти зміну лише перелічених вузлів (paths/schemas/components), решта — інваріант.

6) Політика помилок: кожна операція має узгоджений мінімум відповідей (2xx + 4xx/5xx), єдина схема помилки, опис причин.

7) Людина-у-контурі: якщо breaking change неминучий, LLM має не "вносити" його, а сформувати рекомендацію з планом міграції (версія/депрекейт/soft rollout).

Висновки

Запропоновано процедуру LLM-асистованого рефакторингу OpenAPI-контрактів із guardrails, що знижує ризик невалідних правок і прихованих breaking changes. Ключова ідея — звести роль LLM до генерації структурованого патча та пояснюваного плану змін, а контроль якості покласти на формальні перевірки (OpenAPI/JSON Schema), лінтування та автоматичний diff-контроль сумісності. Подальші дослідження доцільно спрямувати на (1) емпіричну валідацію на корпусі реальних контрактів, (2) калібрування індексу повноти під політики організацій, (3) інтеграцію з контрактними тестами й трасуванням викликів для перевірки відповідності “контракт ↔ реалізація”.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. OpenAPI Initiative. OpenAPI Specification. Version 3.1.0 [Електронний ресурс]. – 2021. – Режим доступу: https://spec.openapis.org/oas/v3.1.0.html (дата звернення: 19.02.2026).
2. OpenAPI Initiative. OpenAPI Specification 3.1.0 Released [Електронний ресурс]. – 2021. – Режим доступу: https://www.openapis.org/blog/2021/02/18/openapi-specification-3-1-released (дата звернення: 19.02.2026).
3. Wright A., Andrews H., Hutton B., Dennis G. A Media Type for Describing JSON Documents (JSON Schema Core). Draft 2020-12 [Електронний ресурс]. – 2020. – Режим доступу: https://json-schema.org/draft/2020-12/json-schema-core (дата звернення: 19.02.2026).
4. OpenAI. Structured model outputs (Structured Outputs) [Електронний ресурс]. – Режим доступу: https://developers.openai.com/api/docs/guides/structured-outputs/ (дата звернення: 19.02.2026).
5. OpenAI. Function calling (tool calling) [Електронний ресурс]. – Режим доступу: https://developers.openai.com/api/docs/guides/function-calling/ (дата звернення: 19.02.2026).
6. Stoplight. Spectral: OpenAPI Rules (built-in oas ruleset) [Електронний ресурс]. – Режим доступу: https://docs.stoplight.io/docs/spectral/4dec24461f3af-open-api-rules (дата звернення: 19.02.2026).
7. oasdiff. OpenAPI Diff and Breaking Changes [Електронний ресурс]. – Режим доступу: https://github.com/oasdiff/oasdiff (дата звернення: 19.02.2026).
8. Chen M., Tworek J., Jun H. та ін. Evaluating Large Language Models Trained on Code [Електронний ресурс]. – 2021. – Режим доступу: https://arxiv.org/abs/2107.03374 (дата звернення: 19.02.2026).

Борис Юрійович Панасюк - Аспірант кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: boris.panasyuk@gmail.com

Наталя Петрівна Бабюк - Доцент кафедри програмного забезпечення Вінницького національного технічного університету, email: babiuk@vntu.edu.ua

Юрій Валерійович Сторожук - Аспірант кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: stoha27@gmail.com

Borys Y. Panasiuk — PhD student, Department of Software Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: boris.panasyuk@gmail.com

Nataliia P. Babiuk — Associate Professor, Department of Software Engineering, Vinnytsia National Technical University, e-mail: babiuk@vntu.edu.ua

Yurii V. Storozhuk — PhD student, Department of Software Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: stoha27@gmail.com