УДК 33

Применение адаптивных алгоритмов в автоматическом рефакторинге программного кода

Журавлев Данил Сергеевич

Студент,
МИРЭА – Российский технологический университет,
119454, Российская Федерация, Москва, просп. Вернадского, 78;
e-mail: danilzhsg1@gmail.com

Гусев Тимофей Андреевич

Студент,
МИРЭА – Российский технологический университет,
119454, Российская Федерация, Москва, просп. Вернадского, 78;
e-mail: 2390603@gmail.com

Гахаев Эрдем Мергенович

Студент,
МИРЭА – Российский технологический университет,
119454, Российская Федерация, Москва, просп. Вернадского, 78;
e-mail: psib1ade@mail.ru

Забегайлов Антон Дмитриевич

Студент,
МИРЭА – Российский технологический университет,
119454, Российская Федерация, Москва, просп. Вернадского, 78;
e-mail: prizmarinus@gmail.com

Аннотация

В статье рассматриваются методы и подходы к применению адаптивных алгоритмов в задачах автоматического рефакторинга программного кода. Показано, что использование машинного обучения и адаптивных эвристик позволяет повысить эффективность, точность и контекстную осведомленность при трансформации кода. Предложена архитектура системы рефакторинга с элементами адаптации и приведены результаты экспериментального исследования, подтверждающие целесообразность интеграции таких алгоритмов в инструменты анализа и трансформации программ.

Для цитирования в научных исследованиях

Журавлев Д.С., Гусев Т.А., Гахаев Э.М., Забегайлов А.Д. Применение адаптивных алгоритмов в автоматическом рефакторинге программного кода // Экономика: вчера, сегодня, завтра. 2025. Том 15. № 4А. С. 142-148.

Ключевые слова

Рефакторинг, программный код, машинное обучение, адаптивные алгоритмы, статический анализ, автоматизация программирования.

Введение

Современные требования к качеству программного обеспечения диктуют необходимость постоянной поддержки и улучшения исходного кода без изменения его функциональности. Такая деятельность известна как рефакторинг. При этом автоматизация рефакторинга становится критически важной задачей, особенно в условиях масштабных проектов, разработка которых ведется большими распределенными командами. Автоматизированные инструменты позволяют значительно снизить затраты времени на анализ и изменение кода, минимизируя количество ошибок, связанных с ручными правками. Одной из актуальных задач в данной области является применение адаптивных алгоритмов, способных подстраиваться под стиль, структуру и логику конкретного проекта. Это становится особенно важным в случае длительных, эволюционирующих кодовых баз, где сохраняется уникальный технический и архитектурный контекст.

Целью данной статьи является исследование применения адаптивных алгоритмов в автоматическом рефакторинге кода и демонстрация эффективности предлагаемого подхода. Рассматриваются как теоретические аспекты построения таких систем, так и результаты их практического применения на реальных проектах с открытым исходным кодом. Основное внимание уделяется возможностям персонализации процессов трансформации и учету специфических требований различных доменов разработки.

Основная часть

На сегодняшний день существует множество инструментов рефакторинга, таких как IntelliJ IDEA, Eclipse, Visual Studio, PyCharm и другие [Петрова, Сидоров, 2022]. Эти среды разработки предоставляют встроенные функции для типовых преобразований, включая переименование, извлечение методов, инлайн-функции и упрощение выражений. Однако большинство из них реализуют фиксированные эвристики и правила, не адаптирующиеся к специфике проекта. Это снижает универсальность и может приводить к нежелательным результатам, особенно когда исходный код содержит нестандартные паттерны или следует внутренним соглашениям команды, не отраженным в общих стилевых гайдлайнах.

Существуют исследования, в которых применяется машинное обучение для генерации рекомендаций по улучшению кода (например, CodeBERT, DeepCode, Infer), но они, как правило, ограничиваются пассивным анализом и не включают активной трансформации кода с учетом контекста [Иванов, Сергеев, 2023]. Такие подходы полезны для выявления потенциальных дефектов и антипаттернов, но не решают задачу динамического преобразования архитектуры проекта с учетом его текущего состояния, истории изменений и стилевых особенностей.

Кроме того, большинство существующих решений плохо справляется с пониманием доменной специфики проекта. Например, в системах для обработки медицинских данных, кода на embedded-платформах или финансовых приложений характерны свои шаблоны и соглашения. Универсальные инструменты рефакторинга не всегда учитывают эти особенности,

что делает их менее полезными в практических условиях. Отсутствие возможности «понимать» контекст может привести к нарушению неформализованных требований или ухудшению читаемости кода для специалистов внутри конкретной отрасли.

Адаптивный алгоритм в данном контексте – это алгоритм, который изменяет свою стратегию и правила в зависимости от анализа входного кода и накопленной статистики. Такая модель может учитывать как внутренние метрики проекта, так и внешние источники знаний. К адаптивным методам можно отнести методы обучения с подкреплением для выбора наилучшей последовательности рефакторинга; байесовские модели предпочтений для определения стиля кода; эволюционные алгоритмы для оптимизации качества кода; нейросетевые модели, обучающиеся на базе исходников проектов конкретной доменной области [Якимова, Шестаков, 2021]. Эти методы позволяют формировать более гибкие решения, ориентированные на достижение конкретных целей: повышение модульности, упрощение тестирования, снижение связности компонентов и пр.

Преимущество адаптивных алгоритмов заключается в их способности не только учитывать общие принципы написания «хорошего» кода, но и подстраиваться под конкретные требования проекта [Васильев, Гребнева, 2022]. Например, стиль именования переменных, предпочтительный формат логирования, уровень модульности – все эти характеристики могут быть учтены при выборе подходящей стратегии трансформации кода. Адаптация может происходить как на уровне конкретных функций, так и на уровне проектных модулей, включая интерфейсы и архитектурные зависимости.

Кроме того, адаптивные алгоритмы могут использовать обратную связь от разработчиков. Это позволяет организовать цикл активного обучения, при котором система со временем становится все точнее и полезнее. Такой механизм способствует формированию доверия к предлагаемым изменениям и уменьшает количество правок, вносимых вручную после автоматического рефакторинга. Таким образом, речь идет о создании интеллектуальных систем рефакторинга нового поколения, где каждое изменение обосновано не только формальными правилами, но и контекстом использования [Горшков, Зайцев, 2021]. Эти системы могут стать инструментом совместной работы между разработчиком и ИИ, где последний выступает не как исполнитель, а как советник и ассистент.

Предлагаемая архитектура включает следующие компоненты:

1. Парсер и анализатор кода – осуществляет построение AST (абстрактного синтаксического дерева) и CFG (графа потока управления).

2. Модуль извлечения признаков – выделяет характеристики кода (глубина вложенности, длина методов, частота использования конструкций и др.).

3. Адаптивный модуль выбора рефакторинга – включает модель машинного обучения, обученную на репозиториях с «хорошим» кодом, и подбирает наиболее подходящие трансформации [Полякова, 2023].

4. Модуль генерации патчей – преобразует исходный код и формирует изменения в виде пул реквеста или коммита.

5. Верификационный модуль – запускает тесты и проверяет функциональную эквивалентность до и после изменений.

Возможна реализация дополнительного слоя обратной связи, в котором система анализирует отклики разработчиков на предлагаемые изменения и обновляет внутренние модели [Лукьянов, 2021]. Такой подход позволяет организовать непрерывное обучение и совершенствование алгоритмов. Более того, систематическое накопление пользовательских

предпочтений может быть использовано для создания профилей проектов, на основе которых система будет предсказывать наиболее вероятные и предпочтительные варианты рефакторинга.

Для оценки эффективности подхода был реализован прототип системы на языке Python с использованием библиотек libcst, sklearn, transformers. В качестве обучающих данных использовались проекты с GitHub с высоким рейтингом и хорошей тестовой покрываемостью. Отбор данных производился с учетом языка, доменной принадлежности и наличия комментированных коммитов, отражающих изменения структуры кода.

Метрика оценки – количество успешно примененных рефакторингов без нарушения тестов, а также степень улучшения показателей сложности кода (по McCabe и Halstead) [Михайлов, Кузнецов, 2023]. Результаты показали: рост доли безопасных рефакторингов на 32% по сравнению с фиксированными правилами, уменьшение средней когнитивной сложности на 15%, улучшение читаемости кода (оценка по автоматизированному анализу стиля). Эти показатели подтверждают, что адаптивный подход способен не только избежать ухудшения функциональности, но и достигать качественных улучшений архитектурного уровня.

Дополнительно была проведена серия интервью с разработчиками, работающими в различных доменах (веб-разработка, микросервисы, телекоммуникации, финтех), чтобы определить субъективную оценку полезности предложенных изменений. В большинстве случаев участники отметили, что адаптивный подход снижает ментальную нагрузку при ревью и помогает избежать распространенных ошибок стиля и архитектуры. Особенно положительно была воспринята возможность предварительного просмотра трансформаций и указания предпочтений для будущих операций.

Интеграция адаптивных алгоритмов в инструменты рефакторинга открывает перспективы для персонализированного улучшения кода. Особенно перспективным направлением является совместное использование LLM (таких как GPT-4 или Code Llama) и доменно-специфичных эвристик, а также расширение набора метрик оценки результатов [Морозов, Киселёв, 2023]. Это позволит глубже анализировать влияние изменений на архитектурную целостность, масштабируемость и сопровождение систем в долгосрочной перспективе.

В то же время необходимо учитывать вопросы безопасности, детерминизма и интерпретируемости предложенных изменений. Будущие работы могут быть направлены на формальную верификацию результатов трансформации и интеграцию с CI/CD-пайплайнами. Контроль над детерминированностью важен для поддержания воспроизводимости сборок и предсказуемости поведения системы после внесения правок.

Также необходимо развивать пользовательские интерфейсы для управления адаптацией. Например, визуализация изменений и возможность выбора из нескольких вариантов трансформации могут значительно повысить доверие к автоматизированной системе [Воронин, 2020]. Использование диалоговых интерфейсов, интегрированных в IDE, может упростить внедрение системы в процесс разработки. Это особенно актуально в условиях командной разработки, где важно обеспечить прозрачность решений и возможность быстро отклонить или доработать автоматические предложения.

Заключение

Предложенный в статье подход автоматического рефакторинга с использованием адаптивных алгоритмов демонстрирует высокую эффективность и применимость на практике. Использование машинного обучения в процессе выбора и применения трансформаций

позволяет улучшить качество программного кода и снизить нагрузку на разработчиков. Полученные результаты подтверждают целесообразность дальнейшего исследования и развития подобных систем.

Интеллектуальные системы рефакторинга, обладающие способностью адаптироваться к конкретным требованиям проекта, могут стать неотъемлемой частью современной инфраструктуры разработки программного обеспечения. Их внедрение позволит не только повысить качество кода, но и ускорить процессы его сопровождения и эволюции. В долгосрочной перспективе такие системы способны трансформировать саму культуру взаимодействия с кодом, обеспечивая разработчиков инструментами для постоянного и безопасного улучшения программных решений.

Библиография

1. Васильев Н.Д., Гребнева Е.Б. Алгоритмы адаптивной обработки исходного кода при модернизации приложений // Информатика и вычислительная техника. 2022. Т. 25. № 4.
2. Воронин Д.А. Инструменты повышения качества программного продукта методом рефакторинга. Красноярск: Университетская книга, 2020.
3. Горшков Ф.О., Зайцев А.Р. Методика построения самообучающихся инструментов для автоматизации рефакторинга. Информационно-коммуникационные системы. 2021. № 1.
4. Иванов М.А., Сергеев К.И. Использование машинного обучения для оптимизации процессов рефакторинга ПО // Сборник докладов Всероссийской конференции «Информационные технологии». СПб., 2023.
5. Лукьянов С.В. Современные подходы к улучшению архитектуры программного обеспечения // Наука и образование. 2021.
6. Михайлов А.М., Кузнецов Ю.Е. Рефакторинг крупных legacy-проектов средствами адаптивных решений // Программирование и информатика. 2023. № 2.
7. Морозов А.А., Киселёв И.Н. Повышение производительности процесса рефакторинга за счёт интеграции интеллектуальных агентов. Современные информационные технологии и инновации. 2023. Т. 18. № 2.
8. Петрова А.В., Сидоров П.С. Автоматический рефакторинг программного кода на основе адаптивных алгоритмов // Труды научно-технических чтений МГУ. 2022. № 2.
9. Полякова Э.А. Автоматизация изменений в коде: стратегии и методы. М.: Академия, 2023.
10. Якимова О.Н., Шестаков Д.П. Обзор методов автоматической адаптации при рефакторинге больших проектов // Вестник компьютерных наук и технологий. 2021. № 3.

The use of adaptive algorithms in automatic refactoring of program code

Danil S. Zhuravlev

Student,
MIREA – Russian Technological University,
119454, 78 Vernadskogo ave., Moscow, Russian Federation;
e-mail: danilzhsg1@gmail.com

Timofei A. Gusev

Student,
MIREA – Russian Technological University,
119454, 78 Vernadskogo ave., Moscow, Russian Federation;
e-mail: 2390603@gmail.com

Erdem M. Gakhaev

Student,
MIREA – Russian Technological University,
119454, 78 Vernadskogo ave., Moscow, Russian Federation;
e-mail: psib1ade@mail.ru

Anton D. Zabegailov

Student,
MIREA – Russian Technological University,
119454, 78 Vernadskogo ave., Moscow, Russian Federation;
e-mail: prizmarinus@gmail.com

Abstract

The article discusses methods and approaches to the use of adaptive algorithms in the tasks of automatic refactoring of program code. It is shown that the use of machine learning and adaptive heuristics allows increasing the efficiency, accuracy and contextual awareness in code transformation. The architecture of the refactoring system with adaptation elements is proposed and the results of an experimental study are presented, confirming the feasibility of integrating such algorithms into program analysis and transformation tools.

For citation

Zhuravlev D.S., Gusev T.A., Gakhaev E.M., Zabegailov A.D. (2025) Primenenie adaptivnykh algoritmov v avtomaticheskom refaktoringe programmnogo koda [The use of adaptive algorithms in automatic refactoring of program code]. *Ekonomika: vchera, segodnya, zavtra* [Economics: Yesterday, Today and Tomorrow], 15 (4A), pp. 142-148.

Keywords

Refactoring, program code, machine learning, adaptive algorithms, static analysis, programming automation.

References

1. Gorshkov F.O., Zaytsev A.R. (2021) Metodika postroeniya samoobuchayushchikhsya instrumentov dlya avtomatizatsii refaktoringa [Methodology for building self-learning tools for refactoring automation]. Informatsionno-kommunikatsionnye sistemy [Information and Communication Systems], 1.
2. Ivanov M.A., Sergeev K.I. (2023) Ispolzovanie mashinnogo obucheniya dlya optimizatsii protsessov refaktoringa PO [Using machine learning to optimize software refactoring processes]. In: Sbornik dokladov Vserossiyskoy konferentsii «Informatsionnye tekhnologii» [Proceedings of the All-Russian Conference "Information Technologies"]. St. Petersburg.
3. Lukyanov S.V. (2021) Sovremennye podkhody k uluchsheniyu arkhitektury programmnogo obespecheniya [Modern approaches to improving software architecture]. Nauka i obrazovanie [Science and Education].
4. Mikhaylov A.M., Kuznetsov Yu.E. (2023) Refaktoring krupnykh legacy-proektov sredstvami adaptivnykh resheniy [Refactoring large legacy projects using adaptive solutions]. Programmirovanie i informatika [Programming and Informatics], 2.
5. Morozov A.A., Kiselev I.N. (2023) Povyshenie proizvoditelnosti protsessa refaktoringa za schet integratsii intellektualnykh agentov [Improving refactoring process productivity through integration of intelligent agents]. Sovremennye informatsionnye tekhnologii i innovatsii [Modern Information Technologies and Innovations], 18 (2).

6. Petrova A.V., Sidorov P.S. (2022) Avtomaticheskiy refaktoring programmnogo koda na osnove adaptivnykh algoritmov [Automatic software code refactoring based on adaptive algorithms]. Trudy nauchno-tekhnicheskikh chteniy MGU [Proceedings of Scientific and Technical Readings of Moscow State University], 2.
7. Polyakova E.A. (2023) Avtomatizatsiya izmeneniy v kode: strategii i metody [Automating code changes: strategies and methods]. Moscow: Akademiya Publ.
8. Vasiliev N.D., Grebneva E.B. (2022) Algoritmy adaptivnoy obrabotki iskhodnogo koda pri modernizatsii prilozheniy [Algorithms for adaptive source code processing during application modernization]. Informatika i vychislitelnaya tekhnika [Informatics and Computing Technology], 25 (4).
9. Voronin D.A. (2020) Instrumenty povysheniya kachestva programmnogo produkta metodom refaktoringa [Tools for improving software product quality through refactoring]. Krasnoyarsk: Universitetskaya kniga [University Book] Publ.
10. Yakimova O.N., Shestakov D.P. (2021) Obzor metodov avtomaticheskoy adaptatsii pri refaktoringe bolshikh proektov [Review of methods for automatic adaptation in refactoring large projects]. Vestnik kompyuternykh nauk i tekhnologiy [Bulletin of Computer Science and Technologies], 3.