НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
сентябрь–октябрь 2026 Том 26 № 5 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
September–October 2026 Vol. 26 No 5 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

doi: 10.17586/2226-1494-2026-26-5-1018-1025
УДК 004.413

Многокритериальная оптимизация процессов разработки программного обеспечения с использованием больших языковых моделей как ассистентов

Петр Вениаминович Горшков[1]✉, Игорь Александрович Бессмертный[2]

[1] Акционерное общество «Универсальная трейдинговая платформа», Санкт-Петербург, 195269, Российская Федерация
[2] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация

[1] narpetri@itmo.ru✉, https://orcid.org/0009-0008-5173-7981
[2] bessmertny@itmo.ru, https://orcid.org/0000-0001-6711-6399

Аннотация

Введение. Рассмотрена проблема управления применением больших языковых моделей на базе трансформеров в процессах разработки программного обеспечения. Актуальность исследования обусловлена тем, что внедрение больших языковых моделей (Large Language Models, LLM) в разработку программного обеспечения часто сопровождается завышенными ожиданиями: предполагаемое сокращение сроков не всегда достигается, а риски снижения качества, роста затрат, появления уязвимостей и необходимости дополнительной проверки результата сохраняются. Предметом исследования является процесс применения LLM в разработке программного обеспечения как управляемый режим выполнения проектных работ. Научная новизна подхода состоит в представлении применения LLM не как отдельного инструмента генерации кода, а как многокритериальной модели выбора рационального режима разработки. **Метод.** Представлен многокритериальный подход к управлению применением LLM при разработке программного обеспечения с учетом времени, стоимости, качества результата и рисков. В качестве методологической основы использованы анализ и систематизация научных публикаций, формализация факторов, влияющих на эффективность применения LLM, многокритериальное моделирование и принцип Парето-оптимальности. Выделены управляемые параметры сценария применения LLM: уровень декомпозиции задачи, степень проработки проектной документации, качество постановки задачи и промта, а также выбранная языковая модель или инструмент на ее основе. В качестве дополнительных контекстных параметров предложены: квалификация разработчика или команды, язык программирования и технологический стек. **Основные результаты.** Показано, что указанные параметры влияют на время подготовки документации, генерации результата, проверки и исправления кода, а также на стоимость, качество и риски проекта. В эксперименте использовано сценарное моделирование альтернативных режимов применения LLM, позволяющее сравнивать варианты по критериям времени, стоимости, качества и риска. Разработанная модель позволяет выбирать рациональные сценарии применения LLM на множестве Парето с учетом ограничений конкретного проекта разработки программного обеспечения. **Обсуждение.** В отличие от существующих подходов, ориентированных преимущественно на оценку отдельных инструментов искусственного интеллекта, качества генерации или производительности разработчика, предложенная модель рассматривает LLM как элемент управляемого процесса разработки. Практическое применение подхода связано с адаптацией инструментов управления проектами разработки программного обеспечения: средств визуализации и маршрутизации задач, промт-ориентированных шаблонов документации и реестра неформализованных заметок. Это позволяет снизить вариативность результатов генерации и повысить управляемость применения LLM в разработке программного обеспечения.

Ключевые слова

большие языковые модели, разработка программного обеспечения, многокритериальная оптимизация, Парето-оптимальность, управление проектами разработки программного обеспечения, генерация кода

Ссылка для цитирования: Горшков П.В., Бессмертный И.А. Многокритериальная оптимизация процессов разработки программного обеспечения с использованием больших языковых моделей как ассистентов // Научно-технический вестник информационных технологий, механики и оптики. 2026. Т. 26, № 5. С. 1018–1025. doi: 10.17586/2226-1494-2026-26-5-1018-1025

© Горшков П.В., Бессмертный И.А., 2026

Multi-objective optimization of software development processes using large language models as assistants

Petr V. Gorshkov[1]✉, Igor A. Bessmertny[2]

[1] CEO, Joint Stock Company "Universal Trading Platform", Saint Petersburg, 195269, Russian Federation
[2] ITMO University, Saint Petersburg, 197101, Russian Federation

[1] narpetri@itmo.ru✉, https://orcid.org/0009-0008-5173-7981
[2] bessmertny@itmo.ru, https://orcid.org/0000-0001-6711-6399

Abstract

This article examines the problem of managing the use of large language models based on transformer architectures in software development processes. The relevance of the study is that the implementation of Large Language Models (LLMs) in software development is often accompanied by inflated expectations: the expected reduction in development time is not always achieved, while the risks of reduced quality, increased costs, vulnerabilities, and the need for additional verification of the result remain. The subject of this study is the use of LLMs in software development as a managed mode of project work. The scientific novelty of the proposed approach lies in presenting the use of LLMs not as a standalone code generation tool, but as a multi-objective model for selecting a rational development mode. The aim of this study is to develop a multi-objective approach to managing the use of LLMs in software development, taking into account time, cost, result quality, and risks. The methodological basis includes the analysis and systematization of scientific publications, the formalization of factors influencing the effectiveness of LLM use, multi-objective modeling, and the Pareto optimality principle. The study identifies the following controllable parameters of LLM application scenarios: the level of task decomposition, the degree of elaboration of project documentation, the quality of the problem statement and prompt, and the selected language model or a tool based on it. In addition, contextual parameters are identified: developer or team qualification, programming language, and technology stack. The identified parameters are shown to affect the time required for documentation preparation, result generation, code review and correction, as well as the cost, quality, and risks of the project. The experimental part uses scenario modeling of alternative LLM application modes, which allows comparing options according to time, cost, quality, and risk criteria. The proposed model enables the selection of rational LLM application scenarios on the Pareto set, taking into account the constraints of a specific IT project. Unlike existing approaches mainly focusing on evaluating individual AI tools, generation quality, or developer productivity, the proposed model considers LLMs as an element of a managed software development process. The practical application of the approach involves adapting IT project management tools: task visualization and routing tools, prompt-oriented documentation templates, and a registry of informal notes. This reduces the variability of generated results and improves the manageability of LLM use in software development.

Keywords

large language models, software development, multi-objective optimization, Pareto optimality, IT project management, code generation

For citation: Gorshkov P.V., Bessmertny I.A. Multi-objective optimization of software development processes using large language models as assistants. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2026, vol. 26, no. 5, pp. 1018–1025 (in Russian). doi: 10.17586/2226-1494-2026-26-5-1018-1025

Введение

Повышенная конкуренция в сфере разработки программного обеспечения (ПО) требует сокращения сроков реализации проектов при сохранении качества и соблюдении бюджетных ограничений.

Рост популярности искусственного интеллекта (Artificial Intelligence), в частности больших языковых моделей (Large Language Models, LLM) на базе трансформеров, приводит к их внедрению в процессы разработки. Однако между ожиданиями от использования таких моделей и фактическими результатами сохраняется разрыв, способный приводить к нарушению сроков, снижению качества, превышению бюджета и новым рискам.

Актуальность исследования обусловлена необходимостью оптимизации процессов разработки ПО при использовании LLM, поскольку эффективность их применения зависит не только от квалификации разработчиков, но и от организации рабочих процессов.

Предмет исследования

Анализ научных публикаций показал, что формализованный и количественно обоснованный подход к оптимизации процессов разработки ПО с использованием LLM пока не является устоявшимся и общепринятым.

В то же время интерес к применению LLM в разработке ПО быстро растет. По запросу «large language models in software development» в предметной области «Computer Science» количество научных публикаций в базе данных Scopus увеличилось с 3692 в 2021 году до 9668 в 2025 году. Данные на 1 мая 2026 года отражены на рисунке.

Исследования показывают, что код, сгенерированный с использованием LLM, может содержать больше ошибок, чем код, написанный человеком [1]. Сложность современных архитектур также создает ограничения, с которыми LLM справляются не всегда [2]. Даже работающий код может содержать скрытые уязвимости или аномальное поведение, поэтому его валидация и тестирование требуют повышенного внимания [3]. В силу этих ограничений LLM целесообразнее рассматривать как ассистента, а не замену разработчика [4].

В разработке ПО, в зависимости от этапов его создания, LLM применяются для повышения безопасности жизненного цикла при распределении ответственности между человеком и агентом искусственного интеллекта (ИИ) [5], поддержки архитектурных решений [6], ге-

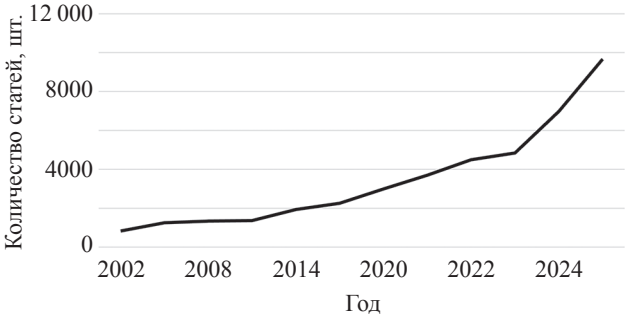

Рисунок. Количество статей в год
Figure. Number of articles per year

нерации кода [7], документирования [8], исправления ошибок, рефакторинга, тестирования [9, 10] и обеспечения безопасности ПО [11]. Более крупные модели обычно показывают более высокое качество результата [2], однако полная замена разработчика возможна только в ограниченных сценариях [12].

Отдельно выделяются риски утечки конфиденциальных данных [13], а также проблемы сохранения и замещения контекста при обучении ИИ-агентов, когда новые данные могут вытеснять старые, а ранее возникшие аномалии — повторно проявляться в дальнейшем [14].

Оценки применения ИИ в разработке неоднозначны: наряду с ростом продуктивности сохраняются опасения, связанные с ограничениями и барьерами интеграции [15, 16]. На текущем этапе ИИ чаще выступает ассистентом, а не полноценной заменой разработчика [17], поскольку сложные проекты пока требуют человеческого контроля [18]. Для широкого применения агентного ИИ также остаются нерешенными вопросы надежности, безопасности, этики и ответственности [19].

Сохраняется разрыв между исследованиями и промышленным применением ИИ-инструментов [14]. Специализированные для написания программного кода модели Code-Specific Large Language Models превосходят универсальные модели в генерации кода, но остаются инструментом разработчика, поскольку проблемы безопасности, утечки данных и контроля результата сохраняются [20]. Применение трансформеров в программной инженерии перспективно, но связано с существенными рисками [21].

В целом современные исследования подтверждают, что ИИ в разработке ПО пока выступает преимущественно как персональный помощник, а не автономная единица [22]. При этом чрезмерная зависимость от ИИ-инструментов может снижать развитие технических, командных и лидерских навыков, что особенно важно для гибких подходов [23].

Эффективность ИИ-помощника зависит от релевантности модели задаче, качества постановки задачи, опыта разработчика [24] и характера самой задачи, поэтому его ценность не является универсальной [25].

Исследовательская проблема заключается в отсутствии формализованного подхода к управлению применением LLM в разработке ПО, позволяющего учитывать влияние различных факторов на время разработки, стоимость, качество готового продукта и риски.

Гипотеза исследования состоит в том, что представление применения LLM как управляемого режима разработки позволяет формализовать условия их использования и выбирать рациональные сценарии. Основу предлагаемого подхода составляет разграничение параметров применения LLM: управляемых, значения которых выбираются в рамках конкретной задачи (уровень декомпозиции, степень проработки документации, качество постановки задачи и промта, выбор модели), и контекстных, которые заданы условиями проекта и не могут быть произвольно изменены (квалификация разработчика или команды, язык программирования и технологический стек). Оптимизация применения LLM при этом сводится к выбору значений управляемых параметров с учетом заданного контекста.

Целью работы является разработка многокритериального подхода к управлению применением LLM в разработке ПО с учетом времени, стоимости, качества результата и рисков.

В настоящей работе решаются следующие задачи: проанализировать современные подходы к применению LLM в разработке ПО и выявить факторы, влияющие на их эффективность; выделить управляемые и контекстные параметры применения LLM, включая декомпозицию задачи, проработку документации, качество постановки задачи, выбор модели, квалификацию разработчика и технологический стек; формализовать многокритериальную модель оценки сценариев применения LLM и выбора рациональных решений на множестве Парето.

Материалы и методы

Выполнен анализ научных публикаций о применении LLM в разработке ПО в базах данных Scopus, Web of Science и Google Scholar. При анализе учитывались актуальность, соответствие теме, авторитетность издания и признанность автора.

Материалы были проанализированы и систематизированы, что позволило выделить факторы, влияющие на эффективность применения LLM: декомпозицию задачи, качество документации, постановку промта и выбор модели (управляемые параметры), а также квалификацию разработчика и технологический стек (контекстные параметры).

В качестве критериев оценки использовались время разработки, стоимость, качество результата и риски. Для выбора рациональных сценариев применения LLM применялись многокритериальное моделирование и принцип Парето-оптимальности.

На заключительном этапе модель была сопоставлена с инструментами управления проектами разработки ПО: средствами визуализации задач, промт-ориентированными шаблонами документации и реестром неформализованных заметок.

Результаты

Применение LLM в разработке ПО предлагается рассматривать не как отдельный инструмент генерации кода, а как управляемый режим выполнения проектных

работ, влияющий на сроки, стоимость, качество результата и риски.

Управляемые параметры: d — уровень декомпозиции задачи; s — степень проработки проектной документации; p — качество постановки задачи и промта; m — выбранная языковая модель или инструмент на ее основе. Контекстные параметры: k — квалификация разработчика или команды; l — язык программирования и технологический стек.

Контекстные параметры не всегда могут быть изменены в рамках конкретной задачи, однако они влияют на допустимые значения и эффективность управляемых параметров. Квалификация разработчика влияет на выбор уровня декомпозиции, качество постановки задачи и способность проверить результат, а язык программирования и технологический стек ограничивают выбор подходящей модели и сложность последующей валидации. Значения параметров модели при этом не являются фиксированными и определяются условиями применения: характером проекта, составом и квалификацией команды, используемым технологическим стеком.

С учетом указанных параметров общее время разработки может быть представлено следующим образом:

$$T = T_{\text{doc}}(s, k, l) + T_{\text{gen}}(d, p, m, k, l) + \\ + T_{\text{check}}(d, p, m, s, k, l) + T_{\text{fix}}(d, p, m, s, k, l), \quad (1)$$

где T_{doc} — время на подготовку и уточнение документации; T_{gen} — время на получение результата с использованием LLM; T_{check} — время проверки сгенерированного решения; T_{fix} — время на исправление и доработку полученного результата.

Полученное выражение показывает, что сокращение времени генерации не гарантирует сокращения общего времени разработки: слабая декомпозиция, неполная документация, некачественный промт, недостаточная квалификация исполнителя или сложность технологического стека могут увеличить объем проверки и исправлений.

Качество результата предлагается рассматривать как функцию от управляемых и контекстных параметров:

$$Q = f(d, p, m, s, k, l),$$

где качество зависит от возможностей модели, формализации задачи, полноты ограничений, критериев проверки, квалификации разработчика и особенностей используемой технологии.

Стоимость разработки представляется как сумма трудовых затрат и затрат на использование LLM:

$$C = c_1(k) \times T + c_2(m) \times U_m,$$

где $c_1(k)$ — стоимость единицы времени разработки с учетом квалификации исполнителя; $c_2(m)$ — стоимость использования выбранной модели; U_m — объем использования модели, определяемый числом запросов, объемом контекста, расходом токенов или иными метриками:

$$U_m = h(d, p, s, m, k, l).$$

Дополнительно в модель включается показатель риска:

$$R = R_{\text{tech}}(m, d, p, s, k, l) + R_{\text{sec}}(m, s, p, k, l),$$

где R_{tech} — риск технических ошибок, некорректной генерации и дополнительных исправлений; R_{sec} — риски безопасности, конфиденциальности и возможных утечек данных.

Таким образом, задача рационального применения LLM в разработке ПО может быть представлена как многокритериальная задача:

$$\min(T, C, R), \max Q$$

при ограничениях:

$$Q \geq Q_{\min}, C \leq C_{\max}.$$

Вектор управляемых параметров обозначим как

$$\mathbf{x} = (d, s, p, m),$$

а вектор контекстных — в виде:

$$\mathbf{z} = (k, l).$$

При этом выбор модели ограничивается технологическим контекстом:

$$m \in M(l),$$

где $M(l)$ — множество моделей, применимых для выбранного языка программирования и технологического стека.

Для выбора рационального сценария используется принцип Парето-оптимальности. Решение x^* при заданном контексте z считается Парето-оптимальным, если не существует другого решения x, которое не хуже по всем критериям и строго лучше хотя бы по одному:

$$x^* \in P(z) \Leftrightarrow \nexists x: \begin{cases} T(x, z) \leq T(x^*, z), \\ C(x, z) \leq C(x^*, z), \\ R(x, z) \leq R(x^*, z), \\ Q(x, z) \geq Q(x^*, z), \end{cases}$$

при этом хотя бы одно из указанных неравенств выполняется строго.

Апробация представленного подхода

Работоспособность приведенных соотношений проверена на числовом примере. Рассмотрена типовая задача — реализация модуля импорта данных из CSV-файла с валидацией записей и REST-эндпоинтом. Контекст: k — разработчик среднего уровня квалификации; l — Kotlin/Ktor. Приняты $c_1(k) = 1500$ руб./ч, $Q_{\min} = 0{,}85$, $C_{\max} = 25\,000$ руб.; качество Q оценивалось долей пройденных приемочных проверок, компоненты риска заданы в шкале [0; 0,5], поэтому Q и R принимают значения в диапазоне [0; 1].

Сопоставлены три сценария: S_0 — без применения LLM; S_1 — низкий уровень декомпозиции, минимальная документация, базовый промт, модель с меньшей стоимостью обращения Claude Sonnet 5 ($c_2 \approx 306$ руб. за 1 млн токенов, $U_m = 0{,}8$ млн); S_2 — средний уровень декомпозиции, проработанная документация, структурированный промт, модель Claude Opus 5 ($c_2 \approx 766$ руб. за 1 млн токенов, $U_m = 2{,}5$ млн). Значения c_2 получены из опубликованных тарифов на обращение к моделям при соотношении входных и выходных токенов 4:1 и пересчитаны в рубли по курсу Банка России на 20 августа 2026 года (85,13 руб./долл. США). Расчет по приведенным выражениям дал следующие значения:

$$T(S_0) = 1{,}0 + 12{,}0 + 2{,}0 + 1{,}5 = 16{,}5 \text{ ч},$$
$$C(S_0) = 1500 \times 16{,}5 = 24\,750 \text{ руб.},$$
$$Q(S_0) = 0{,}90,\ R(S_0) = 0{,}10 + 0{,}05 = 0{,}15;$$
$$T(S_1) = 0{,}5 + 1{,}5 + 3{,}5 + 6{,}0 = 11{,}5 \text{ ч},$$
$$C(S_1) = 1500 \times 11{,}5 + 306 \times 0{,}8 \approx 17\,495 \text{ руб.},$$
$$Q(S_1) = 0{,}72,\ R(S_1) = 0{,}35 + 0{,}20 = 0{,}55;$$
$$T(S_2) = 2{,}0 + 1{,}0 + 2{,}5 + 2{,}5 = 8{,}0 \text{ ч},$$
$$C(S_2) = 1500 \times 8{,}0 + 766 \times 2{,}5 \approx 13\,915 \text{ руб.},$$
$$Q(S_2) = 0{,}88,\ R(S_2) = 0{,}18 + 0{,}10 = 0{,}28.$$

Расчет подтвердил правильность выражения (1): в сценарии S_1 время генерации сокращается с 12,0 до 1,5 ч, однако время проверки и исправлений возрастает с 3,5 до 9,5 ч, поэтому общее время уменьшается до 11,5 ч при снижении качества до 0,72. Ограничению $Q \geq Q_{\min}$ сценарий S_1 не удовлетворяет и в допустимое множество не входит. Сценарии S_0 и S_2 допустимы и не доминируют друг над другом: S_2 предпочтительнее по времени и стоимости, S_0 — по качеству и риску, поэтому оба принадлежат множеству Парето $P(z)$, а выбор между ними определяется приоритетами проекта.

Практическая реализация модели предполагает ее включение в инструментарий управления проектами разработки ПО, обеспечивающий визуализацию задач, фиксацию проектного контекста и контроль результатов генерации.

Обсуждение

Существующие многокритериальные модели учитывают формализацию задач [26], минимизацию времени тестирования при обеспечении надежности [27], множество Парето [28] и специфику отдельных методологий, например SCRUM [29], однако не отражают особенности применения LLM.

Исследования LLM в основном рассматривают качество постановки задачи, обучающие данные, масштаб и архитектуру модели, ее настройки и оценку результата [30–32]. Эмпирические работы показывают неоднозначность эффекта ИИ-инструментов: возможное ускорение зависит от сложности задачи, языка программирования, опыта разработчика и качества взаимодействия с моделью [33–36].

Отдельно выделяются риски уязвимостей, ложной уверенности в результате и необходимости последующей проверки кода [37, 38]. Работы по агентному ИИ также подчеркивают значимость контекста, ролей участников, итеративности промтов и характера совместной работы [39].

В отличие от указанных подходов, предложенная модель рассматривает LLM как управляемый режим разработки: параметры d, s, p, m, k, l задают условия применения модели, а критерии T, C, Q, R позволяют выбирать рациональное решение на множестве Парето с учетом ограничений конкретного проекта разработки ПО.

Заключение

Полученная модель позволяет перейти от общего тезиса о полезности больших языковых моделей к выбору рационального режима их применения. Для каждой задачи может быть определено сочетание декомпозиции, документации, промта и модели с учетом квалификации исполнителя и технологического стека, обеспечивающее приемлемый компромисс между временем, стоимостью, качеством и рисками. Числовой пример подтвердил работоспособность модели: сценарий с минимальной подготовкой сокращает время до 11,5 ч и не удовлетворяет ограничению по качеству, тогда как сбалансированный сценарий обеспечивает 8,0 ч против 16,5 ч и 13 915 руб. против 24 750 руб. при сопоставимом качестве. Для практического применения подхода инструментарий управления проектами разработки программного обеспечения дополняется средствами визуализации и маршрутизации задач, промт-ориентированными шаблонами документации и реестром неформализованных заметок. В средствах визуализации для каждой стадии задачи фиксируются ответственные роли, допустимые действия и обязательные артефакты: требования, критерии приемки, спецификации, диаграммы, отчеты и иные материалы, необходимые для постановки задачи перед большими языковыми моделями и проверки результата. Промт-ориентированные шаблоны задают контекст, ограничения, входные данные, ожидаемый формат результата и критерии качества. Реестр заметок позволяет сохранять слабо формализованные задачи и замечания до их преобразования в полноценные задачи. Разработанный инструментарий снижает вариативность генерации и позволяет управлять применением больших языковых моделей через декомпозицию задач, качество документации, постановку промта, выбор модели и контроль результата.

Литература

1. Bui T.-D., Vu T.T., Nguyen T.-T., Nguyen S., Vo H.D. Correctness assessment of code generated by Large Language Models using internal representations // Journal of Systems and Software. 2025. V. 230. P. 112570. doi: 10.1016/j.jss.2025.112570
2. Yadav D., Mondal S. Evaluating pre-trained Large Language Models on zero shot prompts for parallelization of source code // Journal of Systems and Software. 2025. V. 230. P. 112543. doi: 10.1016/j.jss.2025.112543
3. Glas M., Nirschl C., Lanyado B., van Niekerk J. Insecure by design? A human-centric security perspective on AI-assisted software development // Computers & Security. 2026. V. 164. P. 104842. doi: 10.1016/j.cose.2026.104842
4. Haque Md.A. LLMs: A game-changer for software engineers? // BenchCouncil Transactions on Benchmarks, Standards and Evaluations. 2025. V. 5. N 1. P. 100204. doi: 10.1016/j.tbench.2025.100204
5. Ayouni S., Khan R.A., Maddeh M., Alwageed H.S., Keshta I., Almagrabi A.O. Exploring the synergistic collaboration of Human Agentic-AI in enhancing the security of the software development lifecycle // Egyptian Informatics Journal. 2026. V. 33. P. 100915. doi: 10.1016/j.eij.2026.100915
6. Esposito M., Li X., Moreschini S., Ahmad N., Cerny T., Vaidhyanathan K., et al. Generative AI for software architecture. Applications, challenges, and future directions // Journal of Systems and Software. 2026. V. 231. P. 112607. doi: 10.1016/j.jss.2025.112607
7. Shao D., Ishengoma F. Empirical analysis of generative AI tool adoption in software development // Information and Software Technology. 2026. V. 192. P. 108036. doi: 10.1016/j.infsof.2026.108036
8. Moradi M., Yan K., Colwell D., Samwald M., Asgari R. A Critical review of methods and challenges in Large Language Models // Computers, Materials and Continua. 2025. V. 82. N 2. P. 1681–1698. doi: 10.32604/cmc.2025.061263
9. Sergeyuk A., Golubev Y., Bryksin T., Ahmed I. Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward // Information and Software Technology. 2025. V. 178. P. 107610. doi: 10.1016/j.infsof.2024.107610
10. Amalfitano D., Coppola R., Distante D., Ricca F. AI in GUI-based testing: A survey of techniques, tools, and perceived advantages and limitations // Journal of Systems and Software. 2026. V. 235. P. 112751. doi: 10.1016/j.jss.2025.112751
11. Jelodar H., Bai S., Hamedi P., Mohammadian H., Razavi-Far R., Ghorbani A. Large language model (LLM) for software security: Code analysis, malware analysis, reverse engineering // Journal of Information Security and Applications. 2026. V. 98. P. 104390. doi: 10.1016/j.jisa.2026.104390
12. Malamas N., Tsardoulias E., Panayiotou K., Symeonidis A.L. Toward efficient vibe coding: An LLM-based agent for low-code software development // Journal of Computer Languages. 2025. V. 85. P. 101367. doi: 10.1016/j.cola.2025.101367
13. Yao Y., Duan J., Xu K., Cai Y., Sun Z., Zhang Y. A survey on large language model (LLM) security and privacy: The Good, The Bad, and The Ugly // High-Confidence Computing. 2024. V. 4. N 2. P. 100211. doi: 10.1016/j.hcc.2024.100211
14. Ghaemi H., Alizadehsani Z., Shahraki A., Corchado J. M. Transformers in source code generation: A comprehensive survey // Journal of Systems Architecture. 2024. V. 153. P. 103193. doi: 10.1016/j.sysarc.2024.103193
15. Basha M., Rodríguez-Pérez G. Trust, transparency, and adoption in generative AI for software engineering: Insights from Twitter discourse // Information and Software Technology. 2025. V. 186. P. 107804. doi: 10.1016/j.infsof.2025.107804
16. Banh L., Holldack F., Strobel G. Copiloting the future: how generative AI transforms software engineering // Information and Software Technology. 2025. V. 183. P. 107751. doi: 10.1016/j.infsof.2025.107751
17. Kuhail M.A., Mathew S.S., Khalil A., Berengueres J., Shah S.J.H. “Will I be replaced?” Assessing ChatGPT’s effect on software development and programmer perceptions of AI tools // Science of Computer Programming. 2024. V. 235. P. 103111. doi: 10.1016/j.scico.2024.103111
18. Liu J., Li S., Liu Z., Cheng Z., Guo Y., Guo Y., et al. Towards multi-language repository-level code generation: From-scratch to guided tasks // Neurocomputing. 2026. V. 679. P. 133204. doi: 10.1016/j.neucom.2026.133204

References

1. Bui T.-D., Vu T.T., Nguyen T.-T., Nguyen S., Vo H.D. Correctness assessment of code generated by Large Language Models using internal representations. *Journal of Systems and Software*, 2025. vol. 230, pp. 112570. doi: 10.1016/j.jss.2025.112570
2. Yadav D., Mondal S. Evaluating pre-trained Large Language Models on zero shot prompts for parallelization of source code. *Journal of Systems and Software*, 2025, vol. 230, pp. 112543. doi: 10.1016/j.jss.2025.112543
3. Glas M., Nirschl C., Lanyado B., van Niekerk J. Insecure by design? A human-centric security perspective on AI-assisted software development. *Computers & Security*, 2026, vol. 164, pp. 104842. doi: 10.1016/j.cose.2026.104842
4. Haque Md.A. LLMs: A game-changer for software engineers? *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, 2025, vol. 5, no. 1, pp. 100204. doi: 10.1016/j.tbench.2025.100204
5. Ayouni S., Khan R.A., Maddeh M., Alwageed H.S., Keshta I., Almagrabi A.O. Exploring the synergistic collaboration of Human Agentic-AI in enhancing the security of the software development lifecycle. *Egyptian Informatics Journal*, 2026, vol. 33, pp. 100915. doi: 10.1016/j.eij.2026.100915
6. Esposito M., Li X., Moreschini S., Ahmad N., Cerny T., Vaidhyanathan K., et al. Generative AI for software architecture. Applications, challenges, and future directions. *Journal of Systems and Software*, 2026, vol. 231, pp. 112607. doi: 10.1016/j.jss.2025.112607
7. Shao D., Ishengoma F. Empirical analysis of generative AI tool adoption in software development. *Information and Software Technology*, 2026, vol. 192, pp. 108036. doi: 10.1016/j.infsof.2026.108036
8. Moradi M., Yan K., Colwell D., Samwald M., Asgari R. A Critical review of methods and challenges in Large Language Models. *Computers, Materials and Continua*, 2025, vol. 82, no. 2, pp. 1681–1698. doi: 10.32604/cmc.2025.061263
9. Sergeyuk A., Golubev Y., Bryksin T., Ahmed I. Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward. *Information and Software Technology*, 2025, vol. 178, pp. 107610. doi: 10.1016/j.infsof.2024.107610
10. Amalfitano D., Coppola R., Distante D., Ricca F. AI in GUI-based testing: A survey of techniques, tools, and perceived advantages and limitations. *Journal of Systems and Software*, 2026, vol. 235, pp. 112751. doi: 10.1016/j.jss.2025.112751
11. Jelodar H., Bai S., Hamedi P., Mohammadian H., Razavi-Far R., Ghorbani A. Large language model (LLM) for software security: Code analysis, malware analysis, reverse engineering. *Journal of Information Security and Applications*, 2026, vol. 98, pp. 104390. doi: 10.1016/j.jisa.2026.104390
12. Malamas N., Tsardoulias E., Panayiotou K., Symeonidis A.L. Toward efficient vibe coding: An LLM-based agent for low-code software development. *Journal of Computer Languages*, 2025, vol. 85, pp. 101367. doi: 10.1016/j.cola.2025.101367
13. Yao Y., Duan J., Xu K., Cai Y., Sun Z., Zhang Y. A survey on large language model (LLM) security and privacy: The Good, The Bad, and The Ugly. *High-Confidence Computing*, 2024, vol. 4, no. 2, pp. 100211. doi: 10.1016/j.hcc.2024.100211
14. Ghaemi H., Alizadehsani Z., Shahraki A., Corchado J. M. Transformers in source code generation: A comprehensive survey. *Journal of Systems Architecture*, 2024, vol. 153, pp. 103193. doi: 10.1016/j.sysarc.2024.103193
15. Basha M., Rodríguez-Pérez G. Trust, transparency, and adoption in generative AI for software engineering: Insights from Twitter discourse. *Information and Software Technology*, 2025, vol. 186, pp. 107804. doi: 10.1016/j.infsof.2025.107804
16. Banh L., Holldack F., Strobel G. Copiloting the future: how generative AI transforms software engineering. *Information and Software Technology*, 2025, vol. 183, pp. 107751. doi: 10.1016/j.infsof.2025.107751
17. Kuhail M.A., Mathew S.S., Khalil A., Berengueres J., Shah S.J.H. “Will I be replaced?” Assessing ChatGPT’s effect on software development and programmer perceptions of AI tools. *Science of Computer Programming*, 2024, vol. 235, pp. 103111. doi: 10.1016/j.scico.2024.103111
18. Liu J., Li S., Liu Z., Cheng Z., Guo Y., Guo Y., et al. Towards multi-language repository-level code generation: From-scratch to guided

19. Sapkota R., Roumeliotis K.I., Karkee M. AI Agents vs. Agentic AI: A Conceptual taxonomy, applications and challenges // Information Fusion. 2026. V. 126. Part B. P. 103599. doi: 10.1016/j.inffus.2025.103599
20. Umre J., Parihar A.S., Gupta A. CSLLM: Code-Specific Large Language Models—A Survey // Expert Systems with Applications. 2026. V. 308. P. 130991. doi: 10.1016/j.eswa.2025.130991
21. Xiao Y., Zuo X., Lu X., Dong J.S., Cao X., Beschastnikh I. Promises and perils of using Transformer-based models for SE research // Neural Networks. 2025. V. 184. P. 107067. doi: 10.1016/j.neunet.2024.107067
22. Kemell K.-K., Saarikallio M., Nguyen-Duc A., Abrahamsson P. Still just personal assistants? — A multiple case study of generative AI adoption in software organizations // Information and Software Technology. 2025. V. 186. P. 107805. doi: 10.1016/j.infsof.2025.107805
23. Ahmad M.O., Al-Baik O., Al-haija M.A., Husein A., Morales-Trujillo M. Artificial intelligence learning paradox: how over reliance on it undermines mastery in Agile software project management // Procedia Computer Science. 2026. V. 278. P. 1877–1884. doi: 10.1016/j.procs.2026.03.182
24. Chang T.-S., Huang L.-G. The impact of generative artificial intelligence on software development performance: a study based on the theory of technology dominance // Information & Management. 2026. V. 63. N 3. P. 104306. doi: 10.1016/j.im.2026.104306
25. Stray V., Hanssen G.K., Barbala A., Šmite D., Stol K.-J. What is Generative AI good for? Introduction to the special issue on Generative AI in software engineering // Information and Software Technology. 2025. V. 187. P. 107857. doi: 10.1016/j.infsof.2025.107857
26. Nigar N., Shahzad M.K., Islam S., Oki O., Lukose J.M. Multi-objective dynamic software project scheduling: a novel approach to handle employee's addition // IEEE Access. 2023. V. 11. P. 39792–39806. doi: 10.1109/ACCESS.2023.3265716
27. Fang C.-C., Yeh C.-W. Multi-objective optimization of software testing schedules for modular control Software, considering learning and negligence factors of testing Staffs // Expert Systems with Applications. 2025. V. 279. P. 127359. doi: 10.1016/j.eswa.2025.127359
28. Chen R., Liang C., Gu D., Zhao H. A competence-time-quality scheduling model of multi-skilled staff for IT project portfolio // Computers & Industrial Engineering. 2020. V. 139. P. 106183. doi: 10.1016/j.cie.2019.106183
29. Ozcelikkan N., Tuzkaya G., Alabas-Uslu C., Sennaroglu B. A multi-objective agile project planning model and a comparative meta-heuristic approach // Information and Software Technology. 2022. V. 151. P. 107023. doi: 10.1016/j.infsof.2022.107023
30. Hou X., Zhao Y., Liu Y., Yang Z., Wang K., Li L., et al. Large Language Models for software engineering: a systematic literature review // ACM Transactions on Software Engineering and Methodology. 2024. V. 33. N 8. P. 220. doi: 10.1145/3695988
31. Yang Y., Xia X., Lo D., Grundy J. A survey on deep learning for software engineering // ACM Computing Surveys. 2022. V. 54. N 10s. P. 206. doi: 10.1145/3505243
32. Zan D., Chen B., Zhang F., Lu D., Wu B., Guan B., et al. Large Language Models meet NL2Code: a survey // Proc. of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2023. P. 7443–7464. doi: 10.18653/v1/2023.acl-long.411
33. Vaithilingam P., Zhang T., Glassman E.L. Expectation vs. Experience: evaluating the usability of code generation tools powered by Large Language Models // Proc. of the CHI Conference on Human Factors in Computing Systems Extended Abstracts. 2022. P. 332. doi: 10.1145/3491101.3519665
34. Nguyen N., Nadi S. An empirical evaluation of GitHub copilot's code suggestions // Proc. of the 19th International Conference on Mining Software Repositories. 2022. P. 1–5. doi: 10.1145/3524842.3528470
35. Dakhel A.M., Majdinasab V., Nikanjam A., Khomh F., Desmarais M.C., Jiang Z.M.(Jack). GitHub Copilot AI pair programmer: Asset or Liability? // Journal of Systems and Software. 2023. V. 203. P. 111734. doi: 10.1016/j.jss.2023.111734
36. Ziegler A., Kalliamvakou E., Li X.A., Rice A., Rifkin D., Simister S., et al. Measuring GitHub Copilot's impact on productivity // Communications of the ACM. 2024. V. 67. N 3. P. 54–63. doi: 10.1145/3633453
37. Asare O., Nagappan M., Asokan N. Is GitHub's Copilot as bad as humans at introducing vulnerabilities in code? // Empirical Software tasks. *Neurocomputing*, 2026, vol. 679, pp. 133204. doi: 10.1016/j.neucom.2026.133204
19. Sapkota R., Roumeliotis K.I., Karkee M. AI Agents vs. Agentic AI: A Conceptual taxonomy, applications and challenges. *Information Fusion*, 2026, vol. 126, part B, pp. 103599. doi: 10.1016/j.inffus.2025.103599
20. Umre J., Parihar A.S., Gupta A. CSLLM: Code-Specific Large Language Models—A Survey. *Expert Systems with Applications*, 2026, vol. 308, pp. 130991. doi: 10.1016/j.eswa.2025.130991
21. Xiao Y., Zuo X., Lu X., Dong J.S., Cao X., Beschastnikh I. Promises and perils of using Transformer-based models for SE research. *Neural Networks*, 2025, vol. 184, pp. 107067. doi: 10.1016/j.neunet.2024.107067
22. Kemell K.-K., Saarikallio M., Nguyen-Duc A., Abrahamsson P. Still just personal assistants? — A multiple case study of generative AI adoption in software organizations. *Information and Software Technology*, 2025, vol. 186, pp. 107805. doi: 10.1016/j.infsof.2025.107805
23. Ahmad M.O., Al-Baik O., Al-haija M.A., Husein A., Morales-Trujillo M. Artificial intelligence learning paradox: how over reliance on it undermines mastery in Agile software project management. *Procedia Computer Science*, 2026, vol. 278, pp. 1877–1884. doi: 10.1016/j.procs.2026.03.182
24. Chang T.-S., Huang L.-G. The impact of generative artificial intelligence on software development performance: a study based on the theory of technology dominance. *Information & Management*, 2026, vol. 63, no. 3, pp. 104306. doi: 10.1016/j.im.2026.104306
25. Stray V., Hanssen G.K., Barbala A., Šmite D., Stol K.-J. What is Generative AI good for? Introduction to the special issue on Generative AI in software engineering. *Information and Software Technology*, 2025, vol. 187, pp. 107857. doi: 10.1016/j.infsof.2025.107857
26. Nigar N., Shahzad M.K., Islam S., Oki O., Lukose J.M. Multi-objective dynamic software project scheduling: a novel approach to handle employee's addition. *IEEE Access*, 2023, vol. 11, pp. 39792–39806. doi: 10.1109/ACCESS.2023.3265716
27. Fang C.-C., Yeh C.-W. Multi-objective optimization of software testing schedules for modular control Software, considering learning and negligence factors of testing Staffs. *Expert Systems with Applications*, 2025, vol. 279, pp. 127359. doi: 10.1016/j.eswa.2025.127359
28. Chen R., Liang C., Gu D., Zhao H. A competence-time-quality scheduling model of multi-skilled staff for IT project portfolio. *Computers & Industrial Engineering*, 2020, vol. 139, pp. 106183. doi: 10.1016/j.cie.2019.106183
29. Ozcelikkan N., Tuzkaya G., Alabas-Uslu C., Sennaroglu B. A multi-objective agile project planning model and a comparative meta-heuristic approach. *Information and Software Technology*, 2022, vol. 151, pp. 107023. doi: 10.1016/j.infsof.2022.107023
30. Hou X., Zhao Y., Liu Y., Yang Z., Wang K., Li L., et al. Large Language Models for software engineering: a systematic literature review. *ACM Transactions on Software Engineering and Methodology*, 2024, vol. 33, no. 8, pp. 220. doi: 10.1145/3695988
31. Yang Y., Xia X., Lo D., Grundy J. A survey on deep learning for software engineering. *ACM Computing Surveys*, 2022, vol. 54, no. 10s, pp. 206. doi: 10.1145/3505243
32. Zan D., Chen B., Zhang F., Lu D., Wu B., Guan B., et al. Large Language Models meet NL2Code: a survey. *Proc. of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 7443–7464. doi: 10.18653/v1/2023.acl-long.411
33. Vaithilingam P., Zhang T., Glassman E.L. Expectation vs. Experience: evaluating the usability of code generation tools powered by Large Language Models. *Proc. of the CHI Conference on Human Factors in Computing Systems Extended Abstracts*, 2022, pp. 332. doi: 10.1145/3491101.3519665
34. Nguyen N., Nadi S. An empirical evaluation of GitHub copilot's code suggestions. *Proc. of the 19th International Conference on Mining Software Repositories*, 2022, pp. 1–5. doi: 10.1145/3524842.3528470
35. Dakhel A.M., Majdinasab V., Nikanjam A., Khomh F., Desmarais M.C., Jiang Z.M.(Jack). GitHub Copilot AI pair programmer: Asset or Liability? *Journal of Systems and Software*, 2023, vol. 203, pp. 111734. doi: 10.1016/j.jss.2023.111734
36. Ziegler A., Kalliamvakou E., Li X.A., Rice A., Rifkin D., Simister S., et al. Measuring GitHub Copilot's impact on productivity. *Communications of the ACM*, 2024, vol. 67, no. 3, pp. 54–63. doi: 10.1145/3633453

Engineering. 2023. V. 28. N 6. P. 129. doi: 10.1007/s10664-023-10380-1
38. Perry N., Srivastava M., Kumar D., Boneh D. Do users write more insecure code with AI assistants? // Proc. of the ACM SIGSAC Conference on Computer and Communications Security. 2023. P. 2785–2799. doi: 10.1145/3576915.3623157
39. Hao H., Hasan K.A., Qin H., Macedo M., Tian Y., Ding S.H.H., et al. An empirical study on developers' shared conversations with ChatGPT in GitHub pull requests and issues // Empirical Software Engineering. 2024. V. 29. N 6. P. 150. doi: 10.1007/s10664-024-10540-x

37. Asare O., Nagappan M., Asokan N. Is GitHub's Copilot as bad as humans at introducing vulnerabilities in code? *Empirical Software Engineering*, 2023, vol. 28, no. 6, pp. 129. doi: 10.1007/s10664-023-10380-1
38. Perry N., Srivastava M., Kumar D., Boneh D. Do users write more insecure code with AI assistants? *Proc. of the ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 2785–2799. doi: 10.1145/3576915.3623157
39. Hao H., Hasan K.A., Qin H., Macedo M., Tian Y., Ding S.H.H., et al. An empirical study on developers' shared conversations with ChatGPT in GitHub pull requests and issues. *Empirical Software Engineering*, 2024, vol. 29, no. 6, pp. 150. doi: 10.1007/s10664-024-10540-x

Авторы

Горшков Петр Вениаминович — генеральный директор, Акционерное общество «Универсальная трейдинговая платформа», Санкт-Петербург, 195269, Российская Федерация, SC 60405574800, https://orcid.org/0009-0008-5173-7981, narpetri@itmo.ru

Бессмертный Игорь Александрович — доктор технических наук, профессор, профессор, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, SC 36661767800, https://orcid.org/0000-0001-6711-6399, bessmertny@itmo.ru

Authors

Petr V. Gorshkov — CEO, Joint Stock Company "Universal Trading Platform", Saint Petersburg, 195269, Russian Federation, SC 60405574800, https://orcid.org/0009-0008-5173-7981, narpetri@itmo.ru

Igor A. Bessmertny — D.Sc., Full Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, SC 36661767800, https://orcid.org/0000-0001-6711-6399, bessmertny@itmo.ru

Статья поступила в редакцию 15.05.2026
Одобрена после рецензирования 22.08.2026
Принята к печати 20.09.2026

Received 15.05.2026
Approved after reviewing 22.08.2026
Accepted 20.09.2026

Работа доступна по лицензии
Creative Commons
«Attribution-NonCommercial»