УДК 004.94 URL: https://ptsj.bmstu.ru/catalog/icec/inf_tech/1004.html

МЕТОДЫ СИНТЕЗА ПРОГРАММНОГО КОДА С ИСПОЛЬЗОВАНИЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

В.А. Чувашов chuvashovva@student.bmstu.ru

МГТУ им. Н.Э. Баумана, Москва, Российская Федерация

Работа посвящена методам синтеза программного кода с применением искусственного интеллекта (ИИ). В ней обобщены актуальные методики исследований, направленных на автоматизацию создания программного кода, предоставляя обзор преимуществ, актуальности и научной новизны в данной области. Освещена значимость использования ИИ для ускорения процесса разработки, повышения качества программ и решения проблемы нехватки квалифицированных специалистов в сфере разработки программного обеспечения. Также рассмотрены современные тенденции и представлен комплексный анализ актуальных подходов. Отмечена их значимость и перспективы для дальнейшего развития в сфере создания программного обеспечения с использованием ИИ.

Ключевые слова: искусственный интеллект, программное обеспечение, программный код, разработка программного обеспечения, автоматизация, нейронные сети

Введение. В последние годы в разработке программного обеспечения (ПО) произошли заметные изменения: искусственный интеллект (ИИ) был успешно интегрирован в уникальные факторы развития, от автоматизации до программного обеспечения, тем самым помогая разработчикам создавать более качественные и «чистые» программные продукты. Он становится все более важным инструментом в разработке ПО. Методы синтеза программного кода, основанные на ИИ, служат ключевым элементом в современной сфере разработки ПО. Эти методы обеспечивают значительные преимущества и, несомненно, повышают эффективность и качество процесса создания программ.

Искусственный интеллект помогает оптимизировать и ускорить процесс проектирования, разработки и внедрения ПО. Суть заключается не в том, чтобы инженеров-разработчиков заменили роботы: скорее, чтобы инструменты на базе ИИ работали в роли ассистентов руководителей проектов, бизнес-аналитиков, программистов и инженеров по тестированию. Благодаря этому специалисты по разработке могут создавать и тестировать части кода быстрее и с меньшими затратами [1]. Таким образом, ИИ может стать важным фактором, который приведет к увеличению производительности программистов и повышению качества продуктов.

 ISSN 2541-8009

Научные исследования в данной области предлагают новаторские подходы, такие как использование глубокого обучения для кодирования и генерации кода, а также сочетание символьного выполнения программ с методами машинного обучения. Также ведутся исследования по автоматическому порождению тестов для проверки сгенерированного кода, что способствует повышению его качества и надежности [2].

Данное исследование представляет собой комплексный обзор современных подходов к созданию программного кода с использованием передовых методов ИИ. Эти аспекты вместе позволяют получить обширное представление о текущем состоянии и потенциале методов синтеза кода с использованием ИИ в разработке ПО. В данном контексте будет рассмотрено значение ИИ в разработке ПО и то, как он меняет способы создания и проверки ПО.

Критерии искусственного интеллекта. Искусственный интеллект — это область науки, которая изучает способы создания программ, способных выполнять задачи, требующие человеческого интеллекта, такие как решение проблем, обучение, понимание языка и творчество [3]. В последние годы ИИ стал широко применяться в различных сферах жизни, включая область программирования и разработки ПО [4].

Разработка ПО с помощью ИИ, AI-augmented Software Engineering — термин, который ввели в Gartner в 2020 г. для описания процесса использования технологий ИИ (например, машинного обучения, обработки естественного языка и др.) и для ускорения циклов разработки приложений и DevOps. Ряд вендоров уже выпустил продукты для такого типа работ. Среди них Codota, Deep Code, Google, Kite, Mendix, Microsoft, OutSystems, Parasoft [5].

Искусственный интеллект может помочь программистам в написании кода, предоставляя им различные методы и модели, которые могут упростить, ускорить и улучшить процесс разработки. Для этого ИИ используют различные методы и модели, которые основаны на математике, логике, статистике и нейробиологии [6]. В данном обзоре будут рассмотрены некоторые из этих методов и моделей, а также их примеры применения.

Существует несколько методов синтеза кода с использованием ИИ, которые можно систематизировать по критериям, представленным на рис. 1 [7].

Источник спецификации определяет желаемый результат синтеза кода. Естественный язык часто используется в этом контексте. Например, GitHub Copilot — это инструмент, который предлагает целые строки или блоки кода на основе естественного языка или комментариев. Он обучен на множестве общедоступных репозиториев и может адаптироваться к стилю и привычкам программиста.

ISSN 2541-8009

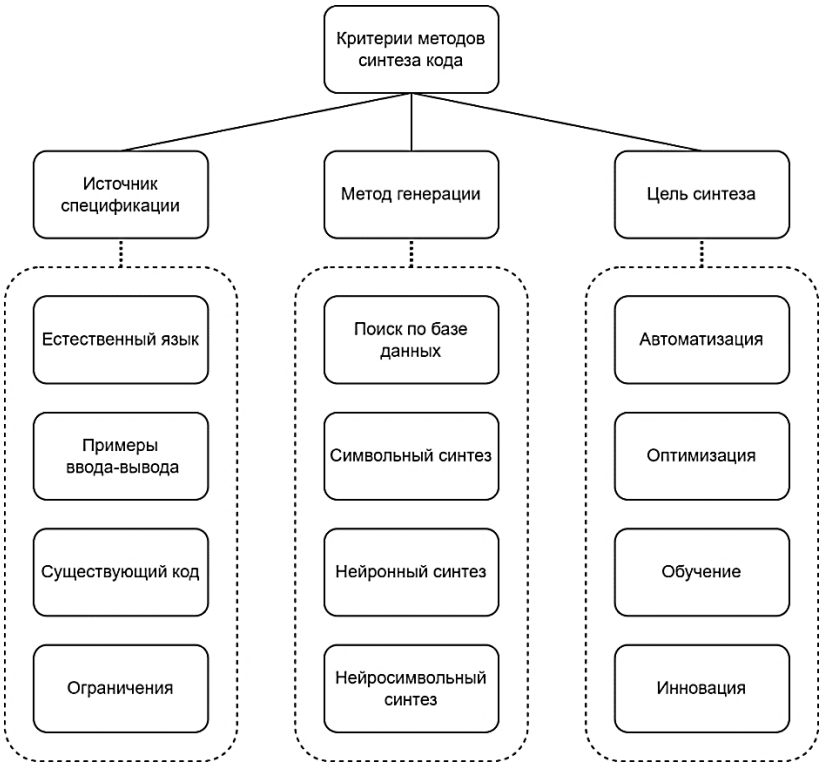

Рис. 1. Критерии методов синтеза кода

Другой способ спецификации — примеры ввода-вывода. Инструмент Replit GhostWriter завершает код в реальном времени, анализируя заданные входные и выходные данные. Он интегрирован с онлайн-редактором кода Replit, что упрощает процесс написания, запуска и отладки будущего программного продукта.

Существуют и системы, такие как SketchAdapt, которые генерируют код на основе заданных пользователем ограничений. Они используют символьный и нейросимвольный синтез для построения и уточнения кандидатов в код. В отличие от создания кода с нуля, многие методы генерации опираются уже на существующий код. Такой сервис, как CodeGuru, анализирует имеющиеся доступные данные и предлагает рекомендации по оптимизации, исправлению ошибок и улучшению качества программы. Также существует метод генерации кода, который включает в себя поиск по базе данных, символьный синтез, нейронный синтез и нейросимвольный синтез (Neuro-Symbolic Program Synthesis). Комбинация перечисленных методов позволяет эффективно и гибко синтезировать код, подстраиваясь под нужды разработчика.

 ISSN 2541-8009

Целью синтеза кода зависит от программиста-разработчика и конкретной поставленной проблемы, в большинстве своем они подразделяются на автоматизацию определенной задачи, обучение принципам программирования, оптимизацию существующего кода или даже инновационные решения сложных проблем.

Автоматизация сгенерированного кода может включать в себя парсинг данных, визуализацию графиков, тестирование программы и многое другое. Обучение кода может демонстрировать принципы программирования (Codeacademy), структуры данных, алгоритмы и паттерны проектирования. Оптимизация кода направлена на улучшение его производительности, надежности, безопасности и читаемости.

Области применения искусственного интеллекта при разработке программного обеспечения. Искусственный интеллект трансформирует сферу разработки ПО, предлагая мощные инструменты и методы, которые повышают эффективность, качество и инновации. Он находит применение в различных аспектах разработки ПО, включая ключевые этапы, такие как сбор технических требований, быстрое прототипирование и кодирование, анализ и обработка ошибок вместе с автоматическим рефакторингом кода, а также тестирование и ввод в эксплуатацию [8, 9].

Цифровые ассистенты могут значительно облегчить процесс разработки, анализируя документы, содержащие требования, и выявляют различия и несоответствия в тексте. Они также обращают внимание на несогласованность в цифрах, единицах измерения и суммах, предлагая возможные решения для исправления этих ошибок. Дальнейшее преобразование требований в программный код обычно занимает месяцы или даже годы. Однако машинное обучение значительно сокращает этот процесс, позволяя специалистам с меньшим опытом использовать методы разработки естественного языка или визуального интерфейса для создания прототипа. В процессе написания кода работающая на базе ИИ система автозаполнения предлагает рекомендации для завершения строчек кода. Интеллектуальные помощники сокращают время на создание кода на 50 %. Дополнительно они могут рекомендовать обратиться к связанным документам, лучшим практикам и дать примеры кода.

Виртуальный ассистент может извлекать уроки из прошлого опыта, чтобы выявлять типичные ошибки и автоматически помечать их на этапе разработки. Машинное обучение можно использовать для анализа системных журналов для быстрого и даже упреждающего выявления ошибок. Чистый код необходим для совместной работы и долгосрочного обслуживания. По мере развития компании программные решения могут изменяться, и остро встает вопрос о том, как модифицировать код для лучшей работы прило-

жений. Машинное обучение используется в этом случае с целью анализа кода и автоматической оптимизации кода для легкой интерпретируемости и повышения производительности. Автоматизированные системы тестирования используют ИИ не только для того, чтобы запускать процесс тестирования, но и для создания тест-кейсов. Иногда ошибки в программном коде становятся явными только после того, как ПО введено в эксплуатацию. Но AI-инструменты предотвращают подобные ситуации, проверяя статистику предыдущих релизов и логи приложений.

Разработка ПО иногда выходит за рамки бюджета и графика. Системы продвинутой аналитики позволяют использовать данные большого количества проектов по разработке ПО для прогнозирования технических задач, необходимых ресурсов и времени на выполнение проекта. Машинное обучение может извлекать данные из прошлых проектов, такие как истории пользователей, определения функций, оценки и фактические условия, для более точного прогнозирования рабочей нагрузки и бюджета.

Инструменты искусственного интеллекта для разработки программного обеспечения. Растущее количество инструментов на базе ИИ поддерживает процессы разработки ПО. Часть из этих решений доступна бесплатно. Ведущие технологические вендоры используют подобные инструменты и предлагают их своим клиентам в виде дополнительных продуктов (plug-in) [10].

Социальная сеть Facebook использует рекомендательный сервис для исправления ошибок и улучшения кода. Последние проекты IBM Mono2Micro и Application Modernization Accelerator (AMA) предоставляют архитекторам приложений инструменты для обновления устаревших приложений и повторного их применения. А Microsoft в 2021 г. объявила, что интегрирует технологии ИИ со своим языком программирования Power Fx, который применяется в разработке приложений на платформе Power Platform [11]. Это позволит клиентам компании создавать программы практически без необходимости написания кода.

В России ИИ активно используется для создания программных продуктов СБЕР. В июле 2021 г. Sber AI зарегистрировала в Роспатенте программу, позволяющую ИИ распознавать и анализировать объекты в виртуальной реальности, следует из материалов ведомства [12].

Согласно опросу американской исследовательской и консалтинговой компании Forrester, 37 % респондентов признали, что они используют ИИ для более эффективного процесса тестирования и разработки [13].

Однако есть и другая сторона у ИИ-продуктов для разработки ПО. Команды, которые используют инструменты для улучшения кода, могут сначала приводить к падению продуктивности, так как ИИ-продукты требуют навы-

 ISSN 2541-8009

ков и умения с ними обращаться [14]. Лишь после глубокого погружения они способны выдавать точные рекомендации для оптимизации разработки ПО.

Использование искусственного интеллекта. Применение ИИ в разработке кода имеет множество примеров. Как было указано выше, он может использоваться для автоматической генерации тестовых данных, оптимизации алгоритмов, анализа и исправления ошибок в коде, а также для предсказания производительности и оптимальных настроек системы.

Исследование производительности нейросети ChatGPT в области программирования выявило интересные результаты, подчеркивающие его успехи и слабые стороны в решении задач. Общий процент успеха составил 71,875 %. Это означает, что ChatGPT успешно предоставил правильные решения, удовлетворяющие тестовым случаям, присутствующим в онлайн-платформе Leetcode. А результирующие данные исследования Johnson et al. показали, что время выполнения кода снизилось на 40 %, а его производительность увеличилась на 25 % [15].

Одной из сильных сторон ChatGPT является его способность эффективно решать структурированные задачи. Исследование показало линейную корреляцию между процентом успеха и процентами принятия задач. Однако выявлены слабости в обработке обратной связи и улучшении решений на основе этой информации [16]. Это указывает на потенциальные ограничения в задачах отладки, где модель испытывает затруднения в улучшении своих решений.

Оценка производительности ChatGPT включала задачи по программированию разной сложности — от простых до сложных и запутанных. Этот подход позволил обеспечить всестороннюю оценку его способностей в решении широкого спектра задач.

Эти результаты говорят о потенциале ChatGPT в области программирования, особенно в структурированных задачах. Однако, для повышения эффективности этой нейросети в задачах отладки и улучшения решений на основе обратной связи требуется дальнейшее исследование и развитие [17, 18].

Использование искусственного интеллекта: преимущества и недостатки. Использование ИИ для генерации кода предлагает ряд преимуществ, которые могут значительно повысить эффективность и качество разработки программного обеспечения [19].

Системы искусственного интеллекта могут быстро генерировать большие объемы кода на основе заданных требований, что существенно сокращает процесс разработки, создавая шаблоны для часто используемых функций или автоматизируя рутинные задачи. Из этого следует и то, что использование такого метода генерации кода может существенно сократить затраты

на разработку ПО за счет ускорения процесса разработки и сокращения необходимости привлечения разработчиков. Способность анализировать большие объемы данных и выявлять закономерности позволяет системам ИИ генерировать более точный и надежный код, содержащий меньшее число синтаксических, а иногда и логических ошибок. Это влечет за собой снижение вероятности сбоев и других проблем с ПО.

Системы искусственного интеллекта могут генерировать новые и творческие решения для проблем программирования, которые могут быть труднодостижимы для разработчика. Например, создание инструментов разработки, использующих ИИ, таких как кодовые редакторы с функцией автозаполнения или системы управления версиями, основанные на предсказаниях ИИ, может значительно повысить производительность разработчиков и качество их кода. Примером такого подхода является TabNine, модель, обученная на огромном количестве открытых исходных кодов, которая предлагает автозаполнение исходя из контекста кода.

Хотя использование ИИ для генерации кода имеет ряд преимуществ, существуют также некоторые ограничения [19]. Одна из ключевых проблем — обеспечение высокого качества сгенерированного кода. Например, модели, основанные на GPT (Generative Pre-trained Transformer — генеративный предобученный трансформер), могут порождать код, который не всегда соответствует требованиям эффективности, безопасности или читаемости. Они могут создавать неоптимальные алгоритмы или структуры данных. Для успешной генерации кода ИИ требуются качественные данные для обучения. Например, различные архитектуры, такие как LSTM (Long Short-Term Memory — разновидность архитектуры рекуррентных нейронных сетей) или модели на основе трансформатора (Transformer-based), используют большие объемы кода для обучения. Недостаток качественных данных может значительно уменьшить точность и качество генерации. В этом же контексте рассматривается и ограниченность обучения. Если обучающие данные недостаточны или непредставительны для всех возможных случаев использования, модель ИИ может выдавать неправильный или неоптимальный код. Если же модель обучалась на ограниченном наборе примеров, она может не учитывать различные стили программирования, архитектурные подходы или специфические требования различных проектов. В случае, когда обучающие данные содержат ошибки, искажения или пропуски, модель ИИ может научиться генерировать код с ошибками или неудовлетворительным качеством.

Данные проблемы решаются пополнением и разнообразием обучающих данных. Это способствует улучшению качества модели и позволяет расширить ее способности к генерации разнообразного и качественного кода. Обу-

 ISSN 2541-8009

чение модели на разнообразных стилях программирования поможет ей создавать более универсальный и адаптируемый код. Регулярное обновление модели, включая ее обучение на новых данных и учет обратной связи, позволяет ей постоянно совершенствоваться.

Ограниченное понимание контекста задачи также является проблемой. Например, модели ИИ могут создавать код, который не полностью соответствует функциональным требованиям из-за ограниченного понимания контекста. Продвинутые архитектуры, такие как BERT (Bidirectional Encoder Representations from Transformers — языковая модель, основанная на архитектуре Transformer), стараются учитывать контекст, но все еще могут иметь ограничения.

Сложность управления и отладки сгенерированного кода также представляет вызов. Если модель ИИ создает сложные конструкции кода, это может затруднить работу разработчиков при отладке или поддержке такого кода.

Решение этих проблем требует разработки более продвинутых и адаптивных моделей ИИ, способных учитывать широкий контекст и требования кода. Например, развитие архитектур, таких как Hybrid AI, которые комбинируют методы машинного обучения и правиловые системы для более точной генерации кода с учетом требований безопасности и эффективности.

Применение ИИ в генерации кода также подвержено вопросам безопасности, поскольку автоматически сгенерированный код может содержать уязвимости, которые могут быть использованы злоумышленниками для атак на ПО или систему.

Модели ИИ могут не полностью понимать контекст безопасности или нюансы конкретных уязвимостей при генерации кода. Это может привести к созданию программ с уязвимыми местами. Если обучающие данные не учитывают аспекты безопасности или содержат уязвимости, модель может научиться создавать код, который также будет уязвимым. После генерации кода ИИ необходимо проводить дополнительные проверки безопасности, включая статический и динамический анализ, чтобы выявить и устранить уязвимости или использовать исходные данных, включающие примеры безопасного кода и сценарии атак, которые помогали бы модели понимать контекст безопасности и создавать более надежный код.

Важно включать в процесс разработки ПО экспертов по безопасности, которые могут регулярно проверять и анализировать сгенерированный код на предмет наличия уязвимостей. Например, можно обучить модель на наборах данных с учетом спецификаций безопасности, таких как OWASP Top 10 [20].

Таким образом, можно выделить основные преимущества и недостатки использования ИИ в задаче синтеза программного кода (рис. 2).

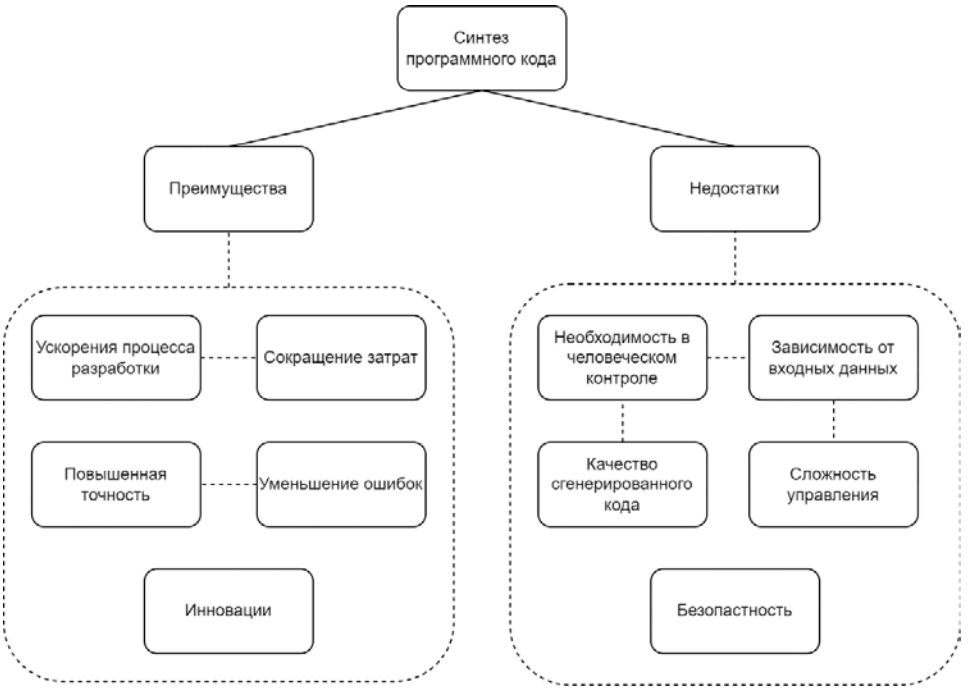

Рис. 2. Преимущества и недостатки использования ИИ

Ожидается, что в будущем использование ИИ для генерации кода получит еще более широкое распространение и развитие. Исследования и разработки в этой области продолжают расширяться, что приведет к созданию более совершенных систем с улучшенными возможностями.

По мере того как ИИ-системы становятся более совершенными, они смогут генерировать все более сложные и надежные коды, что еще больше ускорит процесс разработки ПО и снизит его стоимость. Кроме того, ИИ-системы смогут глубже интегрироваться с другими инструментами и технологиями разработки ПО, что позволит им выполнять более сложные задачи.

Заключение. Внедрение ИИ в процессы разработки кода произвело революцию в разработке ПО. Мощные модели ИИ позволяют автоматизировать генерацию кода, улучшая его эффективность и точность. Однако, чтобы полностью раскрыть потенциал ИИ в разработке кода и свести к минимуму связанные с ним риски, необходимо придерживаться комплексного подхода.

Ключевым элементом является постоянное совершенствование моделей ИИ. Они должны быть обучены на высококачественных данных, представляющих различные аспекты разработки кода. Регулярное переобучение и настройка моделей ИИ гарантируют их актуальность и эффективность.

 ISSN 2541-8009

Кроме того, для обеспечения качества и безопасности кода, генерируемого ИИ, необходимо тщательно его проверять. Тестирование и аудит кода могут выявлять ошибки, уязвимости безопасности и отклонения от стандартов кодирования. Автоматизированные инструменты тестирования могут ускорить этот процесс и улучшить общую надежность кода.

Баланс между автоматизацией и контролем имеет первостепенное значение. Использование ИИ должно дополнять действия разработчиков, а не заменять их. Разработчики должны участвовать в процессе генерации кода, обеспечивая соответствие требованиям и сохранение высокого уровня контроля. Эффективное применение ИИ в генерации кода требует от заинтересованных сторон принятия рациональных решений. Это включает в себя определение подходящих задач для автоматизации с помощью ИИ, предоставление высококачественных данных обучения и регулярную оценку результатов.

По мере того как технология ИИ продолжает развиваться, ее влияние на разработку кода будет только усиливаться. Расширенные возможности генерации кода, проверки и отладки позволят разработчикам создавать более сложные и надежные программные системы с меньшими усилиями и временем.

Интеграция ИИ в генерацию кода открывает беспрецедентные возможности для повышения эффективности разработки ПО, как пример, возможность снижения времени выполнения кода на 40 % и увеличения его производительность на 25 %. Однако для раскрытия этих возможностей необходимо постоянно совершенствовать модели ИИ, проверять генерируемый код и поддерживать баланс между автоматизацией и контролем. Принимая комплексный и последовательный подход, мы можем использовать потенциал ИИ для создания лучшего будущего для разработки ПО.

Литература

[1] Кораблев А.Ю., Булатов Р.Б. Машинное обучение в бизнесе. *Азимут научных исследований: экономика и управление*, 2018, № 2, с. 68–72.

[2] Бевзенко С.А. Применение искусственного интеллекта и машинного обучения в разработке программного обеспечения. *Инновации и инвестиции*, 2023, № 8, с. 187–191.

[3] Sakib F.A., Khan S.H., Karim A.H.M. Extending the frontier of chatGPT: Code generation and debugging. *arXiv preprint arXiv:2307.08260*, 2023.

[4] Becker B.A. et al. Programming is hard-or at least it used to be: educational opportunities and challenges of ai code generation. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, 2023, vol. 1, pp. 500–506. https://doi.org/10.1145/3545945.3569759

[5] Bembenek A. *Combining Datalog and SAT-Based Solving in Code-Reasoning Tools.* Diss. Dr. of Philosophy. Harvard, Harvard University, 2023.

[6] Гаев Л.В., Красиков И.А., Симонов И.Н. Применение искусственного интеллекта в разработке программного обеспечения. *Инновационная наука*, 2023, № 3, с. 45–46.

[7] Петров В.Н. *Веб-разработка: основы и современные технологии*. Санкт-Петербург, БХВ-Петербург, 2019, 760 с.

Бождай А.С., Артамонов Д.В., Евсеева Ю.И. Использование машинного обучения с подкреплением в создании самоадаптивного программного обеспечения. *Известия высших учебных заведений. Поволжский регион. Технические науки*, 2019, № 3, с. 58–68. https://doi.org/10.21685/2072-3059-2019-3-5

[8] Сотников О.О. Смена парадигмы разработки программных продуктов с переходом от человеческого труда на искусственный интеллект. *Аллея науки*, 2020, т. 2, № 5 (44), с. 967–975.

[9] Козлов А.И., Михайлова Е.П. *Программирование веб-приложений на PHP и JavaScript*. Санкт-Петербург, Питер, 2019, 922 с.

[10] Sawant N., Sengamedu S.H. Learning-based identification of coding best practices from software documentation. *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, IEEE, 2022, pp. 533–542. https://doi.org/10.1109/ICSME55016.2022.00073

[11] Цепляев А.Ф. Использование языковых моделей ИИ для изучения програмирования. *Символ науки*, 2023, № 5–2, с. 58–60.

[12] Wermelinger M. Using GitHub Copilot to solve simple programming problems. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, 2023, vol. 1, 2023, pp. 172–178. https://doi.org/10.1145/3545945.3569830

[13] Abadi M., Barham P., Chen J. et al. TensorFlow: A System for Large-Scale Machine Learning. *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*, 2016, vol. 16, pp. 265–283.

[14] Wang Y. et al. GypSum: learning hybrid representations for code summarization. *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*, 2022, pp. 12–23. https://doi.org/10.48550/arXiv.2204.12916

[15] Jiang J. et al. APP-Miner: Detecting API Misuses via Automatically Mining API Path Patterns. *2024 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2024, pp. 4034–4052. https://doi.org/10.1109/SP54263.2024.00043

[16] Barke S., James M.B., Polikarpova N. Grounded copilot: How programmers interact with code-generating models. *CoRR arXiv*, 2022, vol. 2206. https://doi.org/10.48550/arXiv.2206.15000

[17] Pengcheng Y., Graham N. A syntactic neural model for general-purpose code generation. *arXiv preprint arXiv:1704.0169*, 2017. https://doi.org/10.48550/arXiv.1704.01696

 ISSN 2541-8009

[18] Campbell M. Automated coding: The quest to develop programs that write programs. *Computer*, 2020, vol. 53, no. 2, pp. 80–82.
https://doi.org/10.1109/MC.2019.2957958

[19] *Owasp Top 10*. URL: https://owasp.org/www-project-top-ten/ (accessed May 15, 2024).

Поступила в редакцию 04.06.2024

Чувашов Виктор Александрович — бакалавр кафедры «Программное обеспечение ЭВМ и информационные технологии», МГТУ им. Н.Э. Баумана, Москва, Российская Федерация.

Ссылку на эту статью просим оформлять следующим образом:
Чувашов В.А. Методы синтеза программного кода с использованием искусственного интеллекта. *Политехнический молодежный журнал*, 2024, № 05 (94). URL: https://ptsj.bmstu.ru/catalog/icec/inf_tech/1004.html

METHODS OF SYNTHESIZING A PROGRAM CODE USING THE ARTIFICIAL INTELLIGENCE

V.A. Chuvashov chuvashovva@student.bmstu.ru

Bauman Moscow State Technical University, Moscow, Russian Federation

The paper is devoted to the methods of synthesizing a software code using the artificial intelligence (AI). It summarizes the current research methods aimed at automating creation of a software code and provides an overview of advantages, relevance and scientific novelty in this area. The paper highlights the importance of using AI to accelerate the development process, improve the programs quality and solve the problem of shortage of the qualified specialists in software development. Modern trends are also considered, and a comprehensive analysis of current approaches is presented. The paper notes their significance and prospects for further development in software creation using the AI.

Keywords: artificial intelligence, software, program code, software development, automation, neural networks

Received 04.06.2024

Chuvashov V.A. — Bachelor, Department of Computer Software and Information Technologies, Bauman Moscow State Technical University, Moscow, Russian Federation.

Please cite this article in English as:
Chuvashov V.A. Methods of synthesizing a program code using the artificial intelligence. *Politekhnicheskiy molodezhnyy zhurnal*, 2024, no. 05 (94). (In Russ.). URL: https://ptsj.bmstu.ru/catalog/icec/inf_tech/1004.html

 ISSN 2541-8009