Комиссарова Диана Олеговна,
преподаватель
ОГАПОУ «УАвиаК-МЦК»

Турдакова Анна Андреевна,
студент
ОГАПОУ «УАвиаК-МЦК»

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ: АВТОМАТИЗАЦИЯ РУТИННЫХ ЗАДАЧ И УЛУЧШЕНИЕ КАЧЕСТВА КОДА

Аннотация. Статья посвящена применению ИИ и машинного обучения в разработке ПО: автоматизации задач, повышению качества и безопасности кода, а также сокращению сроков и ошибок за счёт интеллектуальных инструментов.

Ключевые слова: Искусственный интеллект, разработка программного обеспечения, автоматизация, машинное обучение, генерация кода, тестирование, отладка, GitHub Copilot, DevSecOps.

Разработка программного обеспечения (ПО) в условиях современного рынка требует постоянного повышения скорости выпуска продуктов (time-to-market) и бескомпромиссного качества. Традиционный процесс разработки характеризуется наличием значительного объема рутинных, повторяющихся задач, которые отнимают значительное время у высококвалифицированных специалистов.

Проблема высокой трудоемкости и потенциальной погрешности, связанной с человеческим фактором, стала катализатором для внедрения интеллектуальных систем. Влияние Искусственного Интеллекта на сферу ИТ стало одним из самых обсуждаемых трендов последнего десятилетия. ИИ, моделируя человеческий интеллект, способен обучаться на огромных массивах данных, выявлять сложные закономерности и применять эти знания для решения новых задач.

Цель данной статьи: показать, как именно ИИ трансформирует процессы разработки ПО, обеспечивая ускорение и повышение надежности конечного продукта.

Интеллектуальная автоматизация охватывает весь цикл разработки ПО, начиная с этапа планирования и заканчивая развертыванием и обслуживанием. Использование ИИ позволяет разработчикам тратить меньше времени на механические действия и больше сосредоточиться на креативных и стратегически важных задачах, тем самым повышая общую производительность.

Начальный этап разработки – сбор и анализ технических требований – традиционно является одним из наиболее подверженных ошибкам из-за нестыковок в документации. Цифровые ассистенты, работающие на базе ИИ, способны анализировать документы с собранными требованиями, указывать на разногласия в тексте, нестыковки в цифрах или единицах измерений, а также предлагать возможные решения. ИИ может проанализировать проект и создать четкое описание для каждого компонента, что существенно облегчает понимание системы.

Наиболее заметное и революционное применение ИИ – это интеллектуальная генерация кода. Инструменты на базе ИИ, такие как GitHub Copilot, могут анализировать существующие кодовые базы и предлагать рекомендации для завершения строк, фрагментов кода или даже целых функций на основе контекста и паттернов.

- Сокращение времени разработки: Интеллектуальные помощники способны сократить время на создание кода до 50%.

- Рекомендации и лучшие практики: Системы ИИ могут рекомендовать обращение к связанной документации, предлагать лучшие практики и предоставлять примеры кода, тем самым ускоряя процесс и обеспечивая единообразие стиля.
- Автоматизация рутинных паттернов: ИИ освобождает программиста от повторения одних и тех же повторяющихся строк и рутинных конструкций (например, геттеры, сеттеры, обработка ошибок), которые составляют значительную часть общей кодовой базы.
- Трансляция кода: Инструменты, такие как CodeMorph, используют ИИ для перевода приложения с одного языка программирования на другой, сохраняя при этом логику и функциональность.

Кроме того, ИИ-инструменты, например, Documatic или Mintlify, могут автоматически генерировать документацию по изменениям в коде, классам и функциям. Это критически важно, поскольку ручное ведение документации часто игнорируется или отстает от реального состояния кода, а автоматизация этого процесса обеспечивает доступ к актуальной информации для всех разработчиков.

Внедрение ИИ в процессы тестирования, отладки и анализа кода стало ключевым фактором для обеспечения высокого качества программных продуктов и сокращения количества ошибок (багов) на этапе релиза.

Отладка и поиск ошибок являются одними из наиболее трудоемких этапов разработки. ИИ-системы используют машинное обучение для анализа системных журналов (логов), что позволяет не только быстро, но и упреждающе выявлять ошибки и аномалии.

- Прогнозирование ошибок: Виртуальные ассистенты могут извлекать уроки из прошлого опыта проекта, чтобы выявлять типичные ошибки и автоматически помечать их на раннем этапе разработки.
- Автоматическое исправление: В экспериментальных исследованиях модели машинного обучения, обученные на миллионах примеров ошибок и их исправлений, продемонстрировали способность находить и исправлять ошибки с точностью до 85%.
- Подробный анализ: ИИ позволяет разработчику выбрать определенные строчки кода, в которых он сомневается, и просмотреть их поведение при разных входных условиях.

Инструменты статического анализа кода, усиленные ИИ, выходят за рамки простого синтаксического контроля. Они способны проводить глубокий анализ:

- Архитектуры и логики: Проверка не только синтаксиса, но и анализ логики и производительности кода, выявление анти-паттернов.
- Покрытие тестами: Инструменты, такие как CodiumAI, фокусируются на генерации осмысленных тестов, которые максимально покрывают функциональность кода, обеспечивая его качество.
- Приоритизация: ИИ помогает приоритизировать найденные дефекты, выделяя наиболее критичные для немедленного исправления, что оптимизирует работу команды QA.

Встраивание ИИ в практики DevSecOps позволяет автоматизировать проверку безопасности на ранних этапах. Инструменты статического анализа безопасности кода (SAST), такие как Snyk Code, используют ИИ для обнаружения уязвимостей, сканирования зависимостей и пакетов, а также предлагают автоматические исправления непосредственно в IDE или CI/CD.

- Корреляция уязвимостей (AVC): Технологии AVC на базе ИИ приоритизируют и дают рекомендации по устранению критических уязвимостей.
- Предотвращение угроз: ИИ-платформы аккумулируют данные из различных инструментов тестирования безопасности и предлагают шаблоны безопасного кода, снижая риск внесения уязвимостей.

Рынок предлагает широкий спектр ИИ-инструментов, которые стали незаменимыми помощниками для разработчиков. Список ИИ-инструментов представлен в таблице 1.

Таблица 1

ИИ-инструменты

Инструмент	Основная функция	Применение
GitHub Copilot	Интеллектуальная генерация кода	Автоматическое завершение строк, функций, генерация тестов на основе контекста.
Tabnine / CodeWhisperer	Автозавершение кода	Ускорение процесса кодирования, предложение фрагментов кода.
Snyk Code	Анализ безопасности кода	Поиск уязвимостей в коде и зависимостях, автоматическое предложение исправлений.
CodiumAI	Генерация тестов	Обеспечение качества кода путем автоматического создания осмысленных юнит-тестов.
ChatGPT / Claude	Языковые модели	Решение общих задач, объяснение алгоритмов, рефакторинг, помощь в написании документации.
GigaChat	Российская нейросеть	Генерация кода на популярных языках, помощь в создании скриптов и документации.

Эти инструменты активно интегрируются в среды разработки (IDE), такие как Microsoft Visual Studio, делая их использование максимально комфортным и повсеместным.

Искусственный интеллект кардинально меняет ландшафт разработки программного обеспечения, выступая не просто как автоматизатор, а как интеллектуальный партнер разработчика. ИИ успешно берет на себя наиболее рутинные и времязатратные этапы: от сбора требований и генерации шаблонного кода до автоматического рефакторинга и создания тестов. Главным результатом этого сдвига является значительное повышение качества и надежности кода за счет интеллектуальной отладки, проактивного выявления ошибок и непрерывного анализа уязвимостей.

Внедрение ИИ-подхода позволяет:

1. Сэкономить время и ресурсы, выполняя задачи за секунды, которые ранее занимали часы ручного труда.
2. Повысить точность, исключая человеческий фактор из рутинных операций.
3. Улучшить качество ПО, делая релизы более стабильными, а продукты – более надежными.

Несмотря на вызовы, связанные с необходимостью освоения новых навыков и потенциальной предвзятостью обучающих данных, потенциал ИИ в разработке ПО огромен. Будущие тенденции указывают на еще более глубокую интеграцию, где ИИ будет отвечать за проектирование систем и полный цикл разработки сложных архитектур. Таким образом, ИИ не заменяет разработчика, а многократно расширяет его возможности, позволяя сосредоточиться на инновациях и креативных решениях.

Список литературы:

1. Разработка программного обеспечения с поддержкой ИИ: Полное руководство на 2025 год [Электронный ресурс] / SCAND. – Режим доступа: https://scand.com
2. Области применения ИИ в разработке ПО [Электронный ресурс] / Habr. – Режим доступа: https://habr.com
3. Применение искусственного интеллекта и машинного обучения в разработке программного обеспечения [Электронный ресурс] / КиберЛенинка. – Режим доступа: https://cyberleninka.ru
4. Как автоматизировать рутинные задачи с помощью ИИ? [Электронный ресурс] / Neiroseti.ai. – Режим доступа: https://neiroseti.ai
5. Автоматизация бизнес-процессов с помощью ИИ [Электронный ресурс] / Sber Developer. – Режим доступа: https://developer.sber.ru