

КОМПЬЮТЕРНОЕ ОБЕСПЕЧЕНИЕ И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

COMPUTER ENGINEERING AND SOFTWARE

Научная статья
УДК 004.89
<https://doi.org/10.24143/2072-9502-2025-3-57-69>
EDN RJQMKE

Исследование моделей искусственного интеллекта в аспекте генерации программного кода при разработке студентами сложных программных систем

С. А. Фирсова, В. В. Никулин[✉], С. Д. Шибайкин, В. И. Сафонов, Е. В. Гришин

*Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва,
Саранск, Россия, nikulinvv@mail.ru*

Аннотация. С постоянным развитием компьютерных технологий и увеличением сложности программного обеспечения все более важными становятся вопросы эффективной генерации программного кода. Использование современных моделей и методов искусственного интеллекта (ИИ) для автоматизации процесса создания программного кода является одним из перспективных направлений исследований в области программной инженерии. С появлением и развитием нейронных сетей и глубокого обучения возможности автоматизации процесса создания программного кода значительно расширились. Кроме того, рост сложности программного обеспечения и повышение требований к его качеству и производительности подчеркивают актуальность исследования возможностей применения современных методов ИИ для генерации программного кода. Целью работы является анализ качества применения моделей ИИ в генерации программного кода, предназначенного для разработки сложных программных систем студентами направления подготовки «Программная инженерия». Изучена предметная область, связанная с применением возможностей современных моделей ИИ для генерации программного кода. Рассмотрены различные подходы и технологии ИИ, а также их использование в профессиональной подготовке будущих программных инженеров. Исследование включало организацию запросов к моделям ChatGPT-4o, Copilot, Google-gemini-1.5-pro и YandexGPT и последующий сравнительный анализ сгенерированных указанными моделями программных кодов. Для отправки и последующей обработки запросов было разработано автоматизированное клиент-серверное приложение, использованное в исследовании потенциала моделей ИИ для генерации программного кода. Полученные результаты могут быть полезны для преподавателей и студентов направления подготовки «Программная инженерия», а также для разработчиков программного обеспечения, использующих модели ИИ для автоматизации процессов программирования и улучшения качества профессиональной подготовки.

Ключевые слова: искусственный интеллект, модель искусственного интеллекта, программирование, генерация программного кода, сложная программная система, очередь

Для цитирования: Фирсова С. А., Никулин В. В., Шибайкин С. Д., Сафонов В. И., Гришин Е. В. Исследование моделей искусственного интеллекта в аспекте генерации программного кода при разработке студентами сложных программных систем // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. 2025. № 3. С. 57–69. <https://doi.org/10.24143/2072-9502-2025-3-57-69>. EDN RJQMKE.

Original article

Research on artificial intelligence models in the aspect of software code generation as students development of complex software systems

S. A. Firsova, V. V. Nikulin[✉], S. D. Shibaykin, V. I. Safonov, E. V. Grishin

National Research Ogarev Mordovia State University,
Saransk, Russia, nikulinvv@mail.ru[✉]

Abstract. With the constant development of computer technology and the increasing complexity of software, the issues of efficient generation of program code are becoming more and more important. The use of modern models and methods of artificial intelligence (AI) to automate the process of creating program code is one of the promising areas of research in the field of software engineering. With the advent and development of neural networks and deep learning, the possibilities for automating the process of creating program code have expanded significantly. In addition, the growing complexity of software and the increased requirements for its quality and performance emphasize the relevance of researching the possibilities of using modern AI methods to generate program code. The purpose of the work is to analyze the quality of application of AI models in the generation of program code intended for the development of complex software systems by students of the “Software Engineering” training direction. In the course of the study, the subject area related to the application of the capabilities of modern AI models to generate program code was studied. Various approaches and technologies of AI were considered, as well as their use in the training of future software engineers. The study included the organization of queries to the ChatGPT-4o, Copilot, Google-gemini-1.5-pro, and YandexGPT models and subsequent comparative analysis of the software codes generated by these models. For sending and subsequent processing of requests, an automated client-server application was developed, which was used in the study of the potential of AI models for generating program code. The results obtained can be useful for teachers and students of the direction of training “Software Engineering”, as well as for software developers using AI models to automate programming processes and improve the quality of professional training.

Keywords: artificial intelligence, artificial intelligence model, programming, program code generation, complex software system, queue

For citation: Firsova S. A., Nikulin V. V., Shibaykin S. D., Safonov V. I., Grishin E. V. Research of artificial intelligence models in the aspect of software code generation as students development of complex software systems. *Vestnik of Astrakhan State Technical University. Series: Management, computer science and informatics*. 2025;3:57-69. (In Russ.). <https://doi.org/10.24143/2072-9502-2025-3-57-69>. EDN RJQMKE.

Введение

В современном информационном обществе технологии искусственного интеллекта (ИИ) занимают центральное место в различных сферах деятельности, включая программную инженерию. С постоянным развитием компьютерных технологий и увеличением сложности программного обеспечения (ПО) вопросы эффективной генерации программного кода становятся все более актуальными, а генерация программного кода с использованием ИИ – перспективным направлением проектирования различных приложений [1–3]. Особенно это важно для разработки сложных программных систем, где основное внимание уделяется общей концепции их построения. Сложная программная система – это система, состоящая из множества взаимосвязанных компонентов и модулей, которые взаимодействуют друг с другом для выполнения определенных задач. Они должны обеспечивать высокую производительность, особенно при обработке больших объемов данных или выполнении сложных вычислений. В качестве представителей сложных программных систем можно выделить операционные системы,

системы управления базами данных, корпоративные информационные системы, многопользовательские системы. Разработка подобных систем является непростой задачей, в решение которой большой вклад способны внести модели ИИ, позволяющие генерировать необходимый программный код.

Благодаря развитию интернета и цифровых технологий объемы данных, которые можно использовать для обучения моделей ИИ, постоянно растут. Это открывает новые возможности для создания более точных и эффективных моделей генерации кода, способных адаптироваться к различным сценариям использования и учитывать особенности разрабатываемых программ. Улучшение алгоритмов генерации кода и повышение качества сгенерированного кода становятся важной тенденцией. С появлением новых методов машинного обучения, глубокого обучения и обучения с подкреплением становится возможным создание более сложных и адаптивных моделей, способных генерировать высококачественный программный код с минимальными ошибками и учетом специфики конкретной задачи. Это открывает новые перспективы для применения

генерации кода в различных областях, включая разработку ПО, автоматизацию задач программирования и создание инструментов для повышения производительности разработчиков программных систем.

С появлением новых инструментов и технологий становится все более удобным использование генерации кода в повседневной работе разработчиков. Интеграция генерации кода в среды разработки позволяет разработчикам быстро и эффективно создавать ПО, ускоряя процесс разработки и снижая трудозатраты. Это делает генерацию кода востребованной и доступной для будущих программных инженеров.

Целью данного исследования является оценка качества применения моделей ИИ при генерации программного кода, предназначенного для разработки сложных программных систем студентами направления подготовки «Программная инженерия». Для достижения цели были определены следующие задачи:

- выполнить анализ научной литературы, посвященной технологиям генерации кода с помощью ИИ;
- определить перечень моделей ИИ для исследования их эффективности в аспекте генерирования программного кода;
- разработать автоматизированное клиент-серверное приложение, выполняющее оценивание эффективности исследуемых моделей для генерирования программного кода;
- выполнить анализ и сравнение моделей ИИ на основе разработанного перечня критериев оценивания их эффективности для генерирования программного кода, а также статистических методов.

Для сбора и анализа научной литературы, а также для выявления основных тенденций и направлений актуальных работ по теме исследования использовался систематический обзор литературы. Для исследования генерации программного кода различными моделями ИИ в работе применялись общенаучные методы теоретического исследования, такие как индукция, дедукция, анализ и синтез. Кроме того, для установления закономерностей функционирования рассмотренных в работе моделей ИИ использовались такие методы исследования, как абстрагирование, обобщение, идеализация, аналогия.

Технологии генерации кода с помощью искусственного интеллекта

Искусственный интеллект открывает перед разработчиками новые горизонты в создании программ. Его применение способно ускорить процесс разработки, улучшить качество и расширить возможности ПО.

Существует несколько основных подходов к генерации кода с применением ИИ. Каждый из них

имеет свои уникальные характеристики и преимущества. В числе таких подходов – *символьное выполнение* и *символьная регрессия* [4, 5]. Этот метод позволяет генерировать программный код на основе символьных выражений. Символьное выполнение программы используется для символьной регрессии. Это отличается от традиционного метода, когда программный код создается вручную или с использованием шаблонов. Символьное выполнение и символьная регрессия позволяют создавать программы, которые более гибки и адаптивны к изменениям.

Еще один метод – *генетическое программирование* [6, 7]. В этом подходе используются техники генетического алгоритма для эволюции программы. Создаются новые варианты программного кода на основе естественного отбора и мутаций. Генетическое программирование позволяет создавать программы, которые лучше соответствуют заданным критериям, путем эволюции популяции программ.

Автоматическое программирование [8] основано на использовании алгоритмов и методов для автоматического порождения программного кода или его фрагментов. Это включает в себя различные методы, такие как генерация программ с использованием шаблонов, индуктивное программирование и др. Автоматическое программирование упрощает процесс разработки ПО, ускоряя его создание и уменьшая вероятность ошибок.

Глубокое обучение и нейронные сети также предоставляют мощные инструменты для генерации программного кода [9]. С помощью различных архитектур, таких как рекуррентные нейронные сети (RNN), сверточные нейронные сети (CNN) и трансформеры, можно моделировать зависимости между входными данными и выходным программным кодом. Этот подход позволяет создавать более сложные и адаптивные программы, которые могут адаптироваться к различным условиям и требованиям.

Наконец, методы, основанные на *обучении с подкреплением*, предоставляют еще один подход к генерации программного кода [10]. Обучение с подкреплением рассматривает процесс создания программы как последовательность действий, которые модель должна предпринять, чтобы достичь целевого состояния. Этот подход позволяет модели выбирать действия на основе накопленного опыта и максимизировать некоторую целевую функцию, что делает его особенно эффективным для задач, требующих принятия решений на основе динамического окружения.

Каждый из рассмотренных подходов имеет свои уникальные особенности и преимущества, что позволяет выбрать наиболее подходящий метод в зависимости от конкретной задачи и контекста использования.

Анализ и сравнение моделей искусственного интеллекта

Модели ИИ постоянно эволюционируют, получая обновления практически ежедневно. Доступны как платные, так и бесплатные версии этих моделей. Платные версии, как правило, обладают большими объемами данных и большей функциональностью по сравнению с бесплатными. В данной работе рассмотрены бесплатные и условно-бесплатные версии моделей, что позволяет проводить исследование в равных условиях, поскольку используемые версии являются последними доступными версиями для всех пользователей в сети Интернет.

Анализ моделей ИИ проводился совместно со студентами направления подготовки «Программная инженерия». Текст заданий и выводы по ответам моделей представлялись в таблицах, сопровождаемых ответами моделей ИИ. Проверка полученных с использованием моделей ИИ решений проводилась посредством клиент-серверного приложения, разработанного с участием студентов.

Рассмотрим для примера выполнение исследования одной из задач, актуальных при разработке сложных программных систем, а именно использование очереди [11].

Очереди с приоритетом востребованы при разработке сложных программных систем. Так, в операционных системах планировщики задач используют очереди с приоритетом для управления выполнением процессов. В сетевых протоколах они используются для обработки данных с разными уровнями важности. Очередь с приоритетом – разновидность очередей, в которой у каждого элемента есть свой приоритет обслуживания. Работа очереди с приоритетом обладает рядом особенностей.

Значение элемента, как правило, и определяет его приоритет, т. е. у элемента с самым большим значением самый высокий приоритет. Кроме того, элементы с более высоким приоритетом будут удаляться из очереди раньше, чем элементы с более низким приоритетом, независимо от того, в каком порядке они были добавлены.

В C++ PriorityQueue это структура данных, которая хранит элементы в порядке приоритета. Каждый элемент в PriorityQueue имеет ассоциированный приоритет, и элементы с более высоким приоритетом извлекаются из очереди перед элементами с более низким приоритетом. Эта структура данных часто используется в задачах, где необходимо обрабатывать элементы в определенном порядке, например в алгоритмах поиска. Каждый элемент имеет приоритет, который определяет его порядок в очереди, элементы с более высоким приоритетом будут извлечены раньше. Возможны следующие операции: вставка (добавление элемента в очередь с заданным приоритетом), извлечение (удаление элемента с самым высоким приоритетом из очереди) и просмотр (получение элемента с самым высоким приоритетом без его удаления).

Для исследования моделей ИИ был сформулирован следующий запрос: «Написать контейнер типа priority_queue (очередь с приоритетом) с методами push, top. Организовать обработку исключительных ситуаций». Кроме того, были сформулированы дополнительные условия к запросу. Формулировка запроса к модели ChatGPT-4o, дополнительные условия к запросу, а также результаты выполнения запроса моделью ChatGPT-4o представлены в табл. 1.

Таблица 1

Table 1

Результаты выполнения запроса моделью ChatGPT-4o

The results of a query executed by the model ChatGPT-4o

Формулировка запроса	Сгенерированный код
<p><i>Основное условие</i></p> <p>Написать контейнер типа priority_queue (очередь с приоритетом) с методами push, top. Организовать обработку исключительных ситуаций</p> <p><i>Дополнительные условия</i></p> <p>1. Написать программу на языке программирования C++.</p> <p>2. Генерировать необходимые значения случайным образом. Пользователь вводит только размер очереди.</p> <p>3. Выводить значения для визуального представления.</p>	<pre>#include <iostream> #include <vector> #include <queue> #include <stdexcept> class PriorityQueue {private: priority_queue<int,vector<int>, greater<int>> pq; size_t capacity; public: PriorityQueue(size_t cap) : capacity(cap) {cout<<"Сработал конструктор. Создана новая очередь.\n";} ~PriorityQueue() {cout<<"Сработал деструктор. Очередь удалена.\n";} void push(int value)</pre>

Окончание табл. 1
 Ending the Table 1

Формулировка запроса	Сгенерированный код
Выберите пункт: 1 Введите количество элементов новой очереди: 3 Сработал конструктор. Создана новая очередь Выберите пункт: 2 Добавление элемента: 5 Элемент добавлен Очередь: 5 Выберите пункт: 2 Добавление элемента: 7 Элемент добавлен Очередь: 5 7 Выберите пункт: 2 Добавление элемента: 1 Элемент добавлен Очередь: 1 5 7 Выберите пункт: 2 Добавление элемента: 8 Очередь заполнена Очередь: 1 5 7 Выберите пункт: 3 Верхний элемент очереди: 1 Выберите пункт: 4 Сработал деструктор. Очередь удалена	<pre> {if (pq.size())>=capacity) throw overflow_error("Очередь заполнена"); pq.push(value); cout<<"Элемент добавлен\n"; display();} int top() const {if (pq.empty()) throw underflow_error("Очередь пуста"); return pq.top();} void display() const {priority_queue<int,vector<int>,greater<int>>temp=pq; cout << "Очередь: "; while (!temp.empty()) {cout<<temp.top()<<" "; temp.pop();} cout << "\n";} }; int main() {PriorityQueue* queue = nullptr; int choice, value, size; while (true) {cout << "\n1.Создать новую очередь \n2. Добавить элемент в очередь \n3. Показать верхний элемент \n4. Выйти\n Выберите пункт: "; cin>>choice; try {switch (choice) {case 1: cout<<"Введите количество элементов новой очереди:"; cin >> size; delete queue; queue = new PriorityQueue(size); break; case 2: if (!queue) {cout<<"Создайте очередь!\n"; break;} cout<<"Добавление элемента: "; cin >> value; queue->push(value); break; case 3: if (!queue) {cout<<"Создайте очередь!\n"; break;} cout<<"Верхний элемент: "<<queue->top()<<"\n";break; case 4: delete queue; return 0; default: cout<<"Неверный пункт меню. Попробуйте снова.\n"; } } catch (const exception& e) {cout<<e.what()<<"\n";} } } </pre>

Grigova S. A., Nikulin V. V., Shibaikin S. D., Safonov V. I., Grishin E. V. Research of artificial intelligence models in the aspect of software code generation as students development of complex software systems

Для тестирования моделей ИИ нами был определен перечень критериев, которые являются важными для понимания того, насколько эффективным будет использование исследуемых моделей для генерирования программного кода. Среди них, например, запуск без ошибок сгенерированного программного кода в среде программирования,

присутствие в коде всех необходимых элементов, реализация соответствующих методов и др. Перечень критериев и анализ их соответствия полученным результатам представлены в табл. 2, где «+» соответствует выполнению критерия, при этом в скобках могут быть указаны замечания, «-» – критерий не выполнен.

Таблица 2

Table 2

Выполнение заданных критериев при тестировании моделей ИИ
Fulfilment of criteria when testing AI models

Критерий	ChatGPT-4o	Copilot	Google-gemini-1.5-pro	YandexGPT
1. Код написан на языке программирования C++	+			
2. Код запускается без ошибок	+			
3. Код выдает правильный результат: – присутствует меню из 4-х пунктов	+	+ (при запуске выводится не меню, а запрашивается количество элементов в очереди)	+ (меню выводится без указания номеров пунктов, что затрудняет работу с ним)	+ (при запуске выводится не меню, а запрашивается количество элементов в очереди)
– создание очереди выполняется корректно	+			
– добавление элемента в очередь выполняется корректно	+		–	+
– отображение верхнего элемента выполняется корректно	+			
– выход из программы выполняется корректно	+			
4. Обработка исключений:	+			
– обработка ввода неверного пункта меню	+			
– обработка попыток обращения к пустой очереди	+			
– обработка переполнения очереди	+			
5. Реализация дополнительных условий:	+			
– ввод пользователем только размера очереди	+			
– генерация необходимых значений случайным образом	–	+		–
– вывод значений для визуального представления	+			
6. В коде присутствует класс priority queue	+			
7. В классе имеется конструктор	+			
8. В классе имеется деструктор	+			
9. Реализован метод push()	+		–	+
10. Реализован метод top()	+			
11. Реализован метод display()	+			
12. Присутствует обработчик исключения «Очередь заполнена»	+			
13. Присутствует обработчик исключения «Очередь пуста»	+			
14. Присутствует обработчик исключения «Неверный пункт меню»	+			

Как следует из полученных результатов, все представленные модели ИИ в целом справились с поставленной задачей генерации программного кода, который возможно использовать при разработке программных систем. Однако некоторые критерии ими не были выполнены. В связи с этим решения, полученные с использованием ИИ-моделей, требуют более тщательного изучения.

В рамках проведения исследования также было разработано клиент-серверное приложение, позволяющее автоматически выполнять запросы к раз-

личным моделям ИИ, запускать сгенерированный программный код и проводить его анализ с помощью статического анализатора кода. Серверная часть приложения реализована на языке программирования Python с использованием фреймворка Flask, а клиентская часть разрабатывалась с применением языков HTML и JavaScript. В качестве заданий, для выполнения которых генерировался программный код, рассматривались типовые расчеты, представленные в лабораторных практикумах профильных дисциплин направления подго-

товки «Программная инженерия».

Для работы приложения, выполняющего запросы к моделям ИИ, необходимо заполнить входную форму, представленную на рис. 1, при этом выбирается одна из лабораторных работ учебной дисциплины, ИИ-провайдер и конкретная ИИ-модель, а также язык программирования, на котором необходимо получить результат (в данном случае рассматривались C++, Java, Python).

циплины, ИИ-провайдер и конкретная ИИ-модель, а также язык программирования, на котором необходимо получить результат (в данном случае рассматривались C++, Java, Python).

Рис. 1. Входная форма приложения

Fig. 1. Application Input Form

После заполнения данных формы происходит формирование команды (промпта) для нейросети, сформулированной на естественном языке и содержащей значения, установленные пользователем. Пример сформированного промпта представлен на

рис. 2, где `language` – выбранный язык программирования, `description` – описание задания к лабораторной работе, `data_str` – используемые входные и выходные данные.

Напишите код на языке {language}, реализующий следующее задание: '{description}'. "{data_str}". При использовании русского языка нужно подключить необходимые функции для его вывода. Если тройные кавычки не создаются автоматически, нужно обрамлять ими код с обеих сторон. Для Java имя класса должно быть "Main".

Рис. 2. Пример сформированного промпта

Fig. 2. Example of formed prompt

С помощью функции `send_gpt_request()` сформированный промпт отправляется к выбранному ИИ-провайдеру посредством API, при этом в зави-

симости от указанного провайдера (OpenAI, Google или Anthropic), URL и структура данных запроса меняются, как показано на рис. 3.

```
def send_gpt_request(prompt, provider="openai", model="gpt-3.5-turbo"):
    headers = {"Content-Type": "application/json",
              "Authorization": f"Bearer {API_KEY}",
              "Anthropic-Version": '2023-06-01',}
    data = {}
    if provider.lower() == "openai":
        url = f"{API_BASE_URL}/openai/v1/chat/completions"
        data = {"model": model,
              "messages": [{"role": "user", "content": prompt}]}
    elif provider.lower() == "google":
        # Включаем модель в URL для Google API
        url = f"{API_BASE_URL}/google/v1/models/{model}:generateContent"
        data = {"contents": [{"role": "user", "parts": [{"text": prompt}]}]}
    elif provider.lower() == "anthropic":
        url = f"{API_BASE_URL}/anthropic/v1/messages"
        data = {"model": model,
              "messages": [{"role": "user", "content": prompt}],
              "max_tokens": 1024}
    else:
        raise ValueError("Invalid provider name. Please use 'openai', 'google', or 'anthropic'.")
```

Рис. 3. Отправка промпта ИИ-провайдеру

Fig. 3. Sending prompt to the AI provider

После получения json-ответа модели находится блок, содержащий сгенерированный код, представленный на рис. 4.

```
response = requests.post(url, headers=headers, json=data)
if response.status_code == 200:
    response_dict = response.json()
    print(json.dumps(response_dict, indent=4, ensure_ascii=False))
    # Извлечение текста ответа для OpenAI и Anthropic
    if provider.lower() == "openai":
        response_text = response_dict['choices'][0]['message']['content']
    elif provider.lower() == "anthropic":
        response_text = response_dict['content'][0]['text']
    elif provider.lower() == "google":
        response_text = response_dict['candidates'][0]['content']['parts'][0]['text']
    return response_text
else:
    print(f'Ошибка: {response.status_code}, {response.text}')
    return None
```

Рис. 4. Выделение сгенерированного кода из json-ответа модели

Fig. 4. Extract generated code from model json response

Для выполнения сгенерированного кода использовались следующие технологии:

- встроенный интерпретатор Python;
- компилятор GNU Compiler Collection для C++;
- компилятор javac и интерпретатор java для выполнения Java-кода.

Процессы запускаются с помощью функции run() Python-модуля subprocess, которая позволяет работать с внешними процессами, включая передачу данных в программу и получение ее вывода. На рис. 5 показана реализация функции execute_cpp() для выполнения программного кода в компиляторе GCC.

```
def execute_cpp(code, input_data):
    with open('temp.cpp', 'w', encoding='utf-8') as f:
        f.write(code)
    compile_result = subprocess.run(
        ['g++', 'temp.cpp', '-o', 'temp.exe', '-mconsole'], capture_output=True, text=True)
    if compile_result.returncode != 0:
        return f"Compilation failed:\n{compile_result.stderr}"
    result = subprocess.run(['./temp.exe'], input=input_data, text=True, capture_output=True)
    return result.stdout
```

Рис. 5. Реализация функции execute_cpp()

Fig. 5. Implementing the execute_cpp() function

Дополнительно приложение выполняет статический анализ сгенерированного кода, который проводится без его запуска. Такой подход позволяет обнаружить потенциальные ошибки, недочеты или нарушения стиля, которые могут повлиять на производительность, читаемость и безопасность программы. Для выполнения статического анализа применяются специализированные инструменты,

проверяющие программный код на соответствие стандартам качества, а также выявляющие типичные ошибки программирования. В данном приложении используются 3 инструмента для статического анализа: Cppcheck для C++, Checkstyle для Java и Pylint для Python. На рис. 6 приведена реализация функции check_cpp() для проверки программного кода в анализаторе Cppcheck.

```
def check_cpp(file_path):
    try:
        result = subprocess.run(
            ['cppcheck', '--enable=all', file_path], capture_output=True, text=True)
        if result.returncode != 0:
            print("Errors found during analysis!")
            return result.stdout + result.stderr
    except FileNotFoundError:
        return "Cppcheck is not installed or not found in PATH."
```

Рис. 6. Реализация функции check_cpp()

Fig. 6. Implementing the check_cpp() function

Анализ результатов исследования

В дополнение к ранее рассмотренным моделям ChatGPT-4o, Copilot, Google-gemini-1.5-pro и Yandex GPT круг исследуемых моделей был расширен за счет новых версий от OpenAI (ChatGPT-o1-mini, ChatGPT-4) и Google (Google-gemini-1.5-flash). При тестировании моделей использовались 560 типовых задач из лабораторного практикума профильных дисциплин «Основы программирования», «Дискрет-

ная математика», «Объектно-ориентированное программирование», «Линейная алгебра и аналитическая геометрия» направления подготовки 09.03.04 «Программная инженерия». Результаты тестирования, выполненного в разработанном авторами клиент-серверном приложении, сведенные с помощью подключения облачного сервиса для анализа кода SonarCloud, приведены в табл. 3.

Таблица 3

Table 3

Результаты тестирования сгенерированных программных кодов различными ИИ-моделями

Results of testing generated program codes by various AI models

Модель	Язык	Правильные	Незапущенные	Неправильные	Незначительные ошибки
ChatGPT-o1-mini	Python	556 (99,29 %)	0 (0,00 %)	2 (0,36 %)	2 (0,36 %)
	C++	522 (93,21 %)	1 (0,18 %)	1 (0,18 %)	36 (6,43 %)
	Java	518 (92,50 %)	5 (0,89 %)	4 (0,71 %)	33 (5,89 %)
ChatGPT-4	Python	539 (96,25 %)	3 (0,54 %)	4 (0,71 %)	14 (2,50 %)
	C++	522 (93,21 %)	4 (0,71 %)	5 (0,89 %)	29 (5,18 %)
	Java	503 (89,82 %)	10 (1,79 %)	8 (1,43 %)	39 (6,96 %)
ChatGPT-4o	Python	537 (95,89 %)	2 (0,36 %)	2 (0,36 %)	19 (3,39 %)
	C++	530 (94,64 %)	2 (0,36 %)	3 (0,54 %)	25 (4,46 %)
	Java	502 (89,64 %)	12 (2,14 %)	12 (2,14 %)	34 (6,07 %)
Google-gemini-1.5-pro	Python	430 (76,79 %)	1 (0,18 %)	4 (0,71 %)	125 (22,32 %)
	C++	426 (76,07 %)	2 (0,36 %)	5 (0,89 %)	127 (22,68 %)
	Java	407 (72,68 %)	8 (1,43 %)	12 (2,14 %)	133 (23,75 %)
Google-gemini-1.5-flash	Python	423 (75,54 %)	2 (0,36 %)	6 (1,07 %)	129 (23,04 %)
	C++	412 (73,57 %)	2 (0,36 %)	5 (0,89 %)	141 (25,18 %)
	Java	380 (67,86 %)	15 (2,68 %)	16 (2,86 %)	149 (26,61 %)
YandexGPT	Python	322 (57,50 %)	22 (3,93 %)	16 (2,86 %)	200 (35,71 %)
	C++	318 (56,79 %)	13 (2,32 %)	18 (3,21 %)	211 (37,68 %)
	Java	294 (52,50 %)	25 (4,46 %)	30 (5,36 %)	211 (37,68 %)
Copilot	Python	480 (85,71 %)	3 (0,54 %)	3 (0,54 %)	74 (13,21 %)
	C++	466 (83,21 %)	5 (0,89 %)	6 (1,07 %)	83 (14,82 %)
	Java	453 (80,89 %)	16 (2,86 %)	11 (1,96 %)	80 (14,29 %)

Полученные результаты были проанализированы с применением двухфакторного дисперсионного анализа с последующим попарным сравнением средних значений в градациях с помощью критерия Тьюки, вычисления производились в пакете Statistica Toolbox программного комплекса MATLAB. В качестве факторов были рассмотрены ИИ-модели и языки программирования (Python, C++, Java), на которых тре-

бовалось сгенерировать код для поставленной задачи, параметром-откликом являлось количество правильных ответов.

Из рис. 7 следует, что статистически значимы являются различия средних значений в градациях двухфакторного комплекса по обоим факторам (по ИИ-моделям $p < 1,49e-13$, по языкам программирования $p < 2,27e-6$).

Frigova S. A., Nikulin V. V., Shibaikin S. D., Safonov V. I., Grishin E. V. Research of artificial intelligence models in the aspect of software code generation as students development of complex software systems

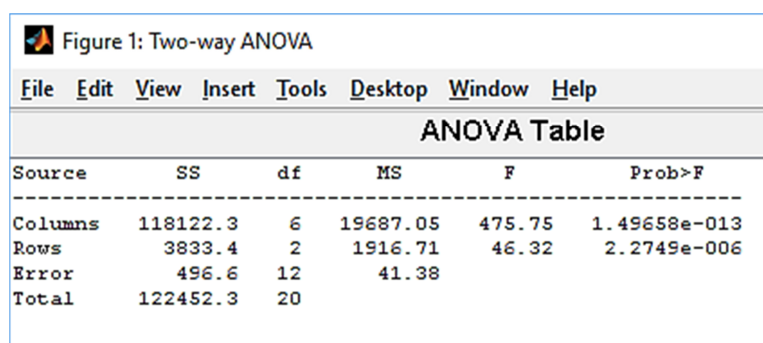
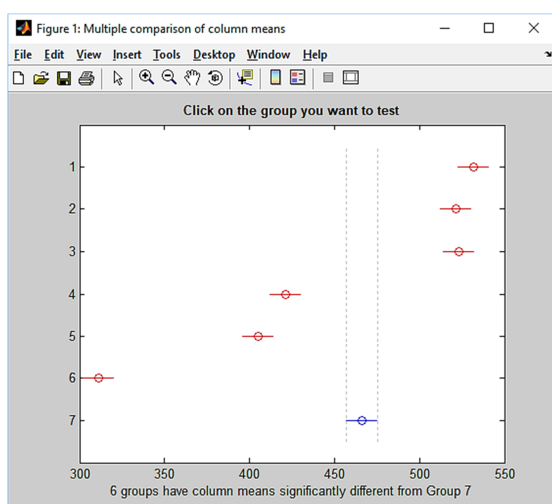


Рис. 7. Дисперсионная таблица двухфакторного анализа

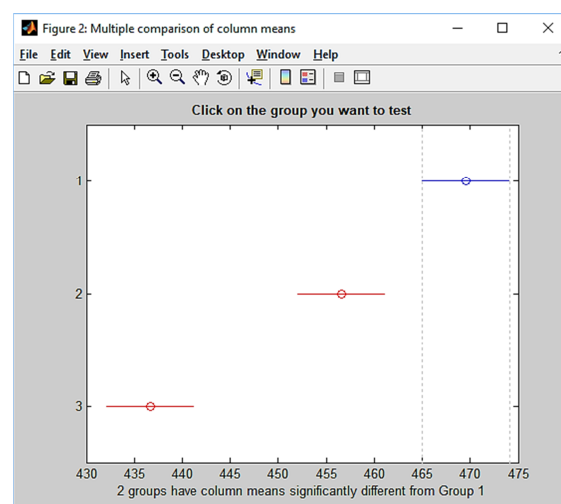
Fig. 7. Two-factor analysis of variance table

Для выявления того, в каких именно градациях средние значения достоверно различаются, были

построены интерактивные графики, приведенные на рис. 8.



a



б

Рис. 8. Интерактивные графики для попарного сравнения средних значений в градациях фактора «ИИ-модели» (*a*); фактора «Языки программирования» (*б*)

Fig. 8. Interactive graphs for pairwise comparison of average values in gradations for the “AI Models” factor (*a*); for the “Programming Languages” factor (*б*)

На рис. 8 видно, что можно достоверно утверждать о различии средних значений правильных ответов между всеми парами языков программирования, а также между всеми парами ИИ-провайдеров: Yandex (YandexGPT), Google (Google-gemini-1.5-pro, Google-gemini-1.5-flash), OpenAI (ChatGPT-o1-mini, ChatGPT-4, ChatGPT-4o), Microsoft (Copilot). Однако различия между ИИ-моделями одного провайдера статистически не доказаны.

В результате можно сделать следующие выводы относительно исследуемых моделей ИИ.

Во-первых, относительно ограничений моделей выявлено следующее:

– YandexGPT: данная модель имеет ограничение на количество символов, равное 1 000. Большинство ответов модели не уместались в ее предел генерируемых ответов, что требовало дополнительных запросов для продолжения генерации ответа. Кроме того, существует ограничение на количество запросов в день, после превышения которого использование модели становится невозможным;

– ChatGPT-4o, ChatGPT-o1-mini, ChatGPT-4: модели имеют ограничение на количество запросов в день. После достижения этого лимита модели автоматически переключаются на более раннюю

версию, которая не имеет ограничений, но генерирует ответы более низкого качества;

– Copilot: у данной модели есть ограничение на количество сообщений в одном диалоге, равное 5. После достижения этого лимита необходимо начинать новый диалог;

– Gemini-1.5-flash, Gemini-1.5-pro: большинство ответов этих моделей также не уместались в установленные ограничения, что требовало дополнительных запросов для продолжения генерации. Кроме того, копирование полных ответов занимает значительное время.

Во-вторых, анализ результатов всех ответов моделей показал следующие результаты производительности данных моделей.

Модели ChatGPT-4o, ChatGPT-o1-mini, ChatGPT-4 продемонстрировали высокий уровень точности, предоставив правильные ответы с незначительными некорректными ошибками, при этом ошибки можно исправить без значительных усилий. Лучшее же показала себя модель ChatGPT-o1-mini.

Модель Copilot также продемонстрировала сравнительно высокий уровень точных ответов с незначительными ошибками. Часть ответов требует небольших исправлений, которые, однако, могут повлиять на окончательное решение, что указывает на необходимость улучшения логики модели.

Модели Gemini-1.5-flash, Gemini-1.5-pro показали средний уровень правильных ответов. Лучшее же показала Gemini-1.5-pro (статистически не доказано).

Модель YandexGPT продемонстрировала низкий уровень правильных ответов. Большая часть

ответов модели оказалась непригодной для использования без значительных изменений.

Таким образом, ИИ-модели провайдера OpenAI продемонстрировали наилучшие результаты среди рассмотренных моделей, предоставив значительное количество правильных ответов с некорректными ошибками.

Заключение

Проведено исследование применения современных моделей ИИ для генерации программного кода при разработке сложных программных систем. Для автоматизации запросов было разработано клиент-серверное приложение. На основании полученных результатов можно сделать вывод, что современные модели ИИ обладают значительным потенциалом для автоматической генерации программного кода, что важно при разработке сложных программных систем. Тем не менее, необходимо продолжить работу над улучшением их точности и надежности. Модель ChatGPT-o1-mini уже сейчас может быть рекомендована для использования в образовательных и профессиональных целях, в то время как другим моделям требуется дополнительная оптимизация. В дальнейшем следует сосредоточиться на развитии данных технологий, что позволит значительно улучшить процесс генерации программного кода и расширить возможности их применения в различных сферах программной инженерии. Обучение будущих программных инженеров применению ИИ-моделей в профессиональной деятельности может значительно повысить эффективность и качество их работы, связанной с созданием сложных программных систем.

Список источников

1. Тимофеев А. Н., Михайлова С. С. Разработка подхода к повышению качества генерации программного кода большими языковыми моделями // *Нейрокомпьютеры: разработка, применение*. 2024. Т. 26. № 4. С. 18–26.
2. Резунин Л., Александров Д. В. Подход к созданию сервиса генерации программного кода мобильных приложений с использованием больших языковых моделей // *ИТ-Стандарт*. 2024. № 4. С. 34–41.
3. Вишнеков А. В., Ерохина Е. А., Иванова Е. М., Трубочкина Н. К. Особенности учебного процесса подготовки ИТ-специалистов в условиях возрастания возможностей генеративного искусственного интеллекта // *Инженерное образование*. 2023. № 34. С. 123–135.
4. Мисонижник А. В., Костюков Ю. О., Костицын М. П., Мордвинов Д. А., Кознов Д. В. Генерация слабейших предусловий программ с динамической памятью в символьном исполнении // *Науч.-техн. вестн. информ. технологий, механики и оптики*. 2022. Т. 22. № 5. С. 982–991.
5. Дивеев А. И., Мендес Флорес Н. Х. Синтез системы пространственной стабилизации мобильного робота

- на основе обучения методом символьной регрессии // *Вестн. Рос. ун-та дружбы народов. Сер.: Инженерные исследования*. 2021. Т. 22. № 2. С. 129–138.
6. Пальмов С. В. Генетические алгоритмы в интеллектуальных системах: учеб. пособие. Самара: Изд-во ПГУТИ, 2021. Ч. 1. 290 с.
7. Крахмалев О. Н. Генетическое программирование и объектное моделирование манипуляционных роботов // *Computational Nanotechnology*. 2023. Т. 10. № 2. С. 16–25.
8. Латин Ю. Э. Современные подходы к автоматическому программированию // *Актуальные исследования*. 2023. № 17. С. 36–39.
9. Широков И. Б., Колесова С. В., Кучеренко В. А., Серебряков М. Ю. Анализ технологий глубокого обучения с подкреплением для систем машинного зрения // *Изв. Тульс. гос. ун-та. Техн. науки*. 2022. № 10. С. 118–120.
10. Саттон Ричард С., Барто Эндрю Г. Обучение с подкреплением. М.: ДМК-Пресс, 2020. 552 с.
11. Павловская Т. А. C/C++. Программирование на языке высокого уровня. СПб.: Питер, 2003. 461 с.

References

1. Timofeev A. N., Mikhailova S. S. Razrabotka podkhoda k povysheniiu kachestva generatsii programmnogo koda bol'shimi iazykovymi modeliami [Developing an approach to improving the quality of program code generation using large language models]. *Neirokom-p'utery: razrabotka, primeneniye*, 2024, vol. 26, no. 4, pp. 18-26.
2. Rezunik L., Aleksandrov D. V. Podkhod k sozdaniyu servisa generatsii programmnogo koda mobil'nykh prilozhenii s ispol'zovaniem bol'shikh iazykovykh modelei [An approach to creating a mobile application code generation service using large language models]. *IT-Standart*, 2024, no. 4, pp. 34-41.
3. Vishnekov A. V., Erokhina E. A., Ivanova E. M., Trubochkina N. K. Osobennosti uchebnogo protsessa podgotovki IT-spetsialistov v usloviyakh vozzrastaniya vozmozhnostei generativnogo iskusstvennogo intellekta [Features of the educational process of training IT specialists in the context of increasing the capabilities of generative artificial intelligence]. *Inzhenernoye obrazovanie*, 2023, no. 34, pp. 123-135.
4. Misonizhnik A. V., Kostiukov Iu. O., Kostitsyn M. P., Mordvinov D. A., Koznov D. V. Generatsiya slabeishikh preduslovii programm s dinamicheskoi pamiat'yu v simvol'nom ispolnenii [Generating the weakest prerequisites for programs with dynamic memory in symbolic execution]. *Nauchno-tekhnicheskii vestnik informatsionnykh tekhnologii, mekhaniki i optiki*, 2022, vol. 22, no. 5, pp. 982-991.
5. Diveev A. I., Mendes Flores N. Kh. Sintez sistemy prostranstvennoi stabilizatsii mobil'nogo robota na osnove obucheniya metodom simvol'noi regressii [Synthesis of a mobile robot spatial stabilization system based on symbolic regression learning]. *Vestnik Rossiiskogo universiteta druzhby narodov. Seriya: Inzhenernye issledovaniya*, 2021, vol. 22, no. 2, pp. 129-138.
6. Pal'mov S. V. *Geneticheskie algoritmy v intellektual'nykh sistemakh: uchebnoye posobie* [Genetic algorithms in intelligent systems: a textbook]. Samara, Izd-vo PGUTI, 2021. Part 1. 290 p.
7. Krakhmalev O. N. Geneticheskoe programmirovaniye i ob'ektnoe modelirovaniye manipulyatsionnykh robotov [Genetic programming and object modeling of manipulation robots]. *Computational Nanotechnology*, 2023, vol. 10, no. 2, pp. 16-25.
8. Latin Iu. E. *Sovremennyye podkhody k avtomaticheskomu programmirovaniyu* [Genetic algorithms in intelligent systems: a textbook]. *Aktual'nye issledovaniya*, 2023, no. 17, pp. 36-39.
9. Shirokov I. B., Kolesova S. V., Kucherenko V. A., Serbriakov M. Iu. Analiz tekhnologii glubokogo obucheniya s podkrepleniem dlia sistem mashinnogo zreniya [Analysis of deep learning technologies with reinforcement for machine vision systems]. *Izvestiya Tul'skogo gosudarstvennogo universiteta. Tekhnicheskie nauki*, 2022, no. 10, pp. 118-120.
10. Sutton Richard S., Barto Endriu G. *Obucheniye s podkrepleniem* [Reinforcement learning]. Moscow, DMC-Press, 2020. 552 p.
11. Pavlovskaya T. A. *C/C++. Programmirovaniye na iazyke vysokogo urovnya* [C/C++. Programming in a high-level language]. Saint Petersburg, Piter Publ., 2003. 461 p.

Статья поступила в редакцию 28.03.2025; одобрена после рецензирования 09.06.2025; принята к публикации 21.07.2025
The article was submitted 28.03.2025; approved after reviewing 09.06.2025; accepted for publication 21.07.2025

Информация об авторах / Information about the authors

Светлана Анатольевна Фирсова – кандидат физико-математических наук, доцент; доцент кафедры информационных технологий и программирования; Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва; karpushkinasa@yandex.ru

Владимир Валерьевич Никулин – кандидат технических наук, доцент; доцент кафедры измерительных и инфокоммуникационных технологий; Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва; nikulinvv@mail.ru

Сергей Дмитриевич Шибайкин – кандидат технических наук, доцент; заведующий кафедрой информационных технологий и программирования; Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва; shibaikinsd@rambler.ru

Svetlana A. Firsova – Candidate of Physico-Mathematical Sciences, Assistant Professor; Assistant Professor of the Department of Information Technology and Programming; National Research Ogarev Mordovia State University; karpushkinasa@yandex.ru

Vladimir V. Nikulin – Candidate of Technical Sciences, Assistant Professor; Assistant Professor of the Department of Measurement and Information and Communication Technologies; National Research Ogarev Mordovia State University; nikulinvv@mail.ru

Sergey D. Shibaykin – Candidate of Technical Sciences, Assistant Professor; Head of the Department of Information Technology and Programming; National Research Ogarev Mordovia State University; shibaikinsd@rambler.ru

Владимир Иванович Сафонов – кандидат физико-математических наук, доцент; доцент кафедры инфокоммуникационных технологий и программирования; Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва; wawans@yandex.ru

Егор Валерьевич Гришин – студент, направление подготовки «Программная инженерия»; Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва; egorgrishin35@gmail.com

Vladimir I. Safonov – Candidate of Physico-Mathematical Sciences, Assistant Professor; Assistant Professor of the Department of Information Technology and Programming; National Research Ogarev Mordovia State University; wawans@yandex.ru

Egor V. Grishin – Student of the training area “Software Engineering”; National Research Ogarev Mordovia State University; egorgrishin35@gmail.com

