frontiers | Frontiers in Computer Science

TYPE Brief Research Report
PUBLISHED 30 December 2024
DOI 10.3389/fcomp.2024.1473870

OPEN ACCESS

EDITED BY
Newton Howard,
Massachusetts Institute of Technology,
United States

REVIEWED BY
Heitor Costa,
Universidade Federal de Lavras, Brazil
Tomasz Górski,
Gdynia Maritime University, Poland

*CORRESPONDENCE
Gulnar Mukhamejanova
✉ gulnar.mukhamedzhanova@narxoz.kz

RECEIVED 11 August 2024
ACCEPTED 04 November 2024
PUBLISHED 30 December 2024

CITATION
Uandykova M, Baitenova L, Mukhamejanova G, Yeleukulova A and Mirkassimova T (2024) Java coding using artificial intelligence.
Front. Comput. Sci. 6:1473870.
doi: 10.3389/fcomp.2024.1473870

Java coding using artificial intelligence

Mafura Uandykova[1], Laura Baitenova[2], Gulnar Mukhamejanova[1]*, Assel Yeleukulova[3] and Tolkyn Mirkassimova[1]

[1]School of Digital Technologies, Narxoz University, Almaty, Kazakhstan, [2]Department of Information Technology, Turan University, Almaty, Kazakhstan, [3]Kazmetengineering LLP, Almaty, Kazakhstan

This study explores the potential of chatbots, specifically ChatGPT, in Java software development. The aim is to classify tasks for effective use of industrial code and develop recommendations for applying chatbot assistance, identifying boundaries where human intervention remains essential. The methodology included analyzing scientific literature and empirically testing ChatGPT-3.5 on various Java development tasks. The tasks were divided into simple (working with XML, JSON, multithreading, and data input/output) and complex (writing MVC applications, REST services, and GUI). The results showed that ChatGPT successfully handles simple tasks but struggles with complex problems. The study identified scenarios where the chatbot can effectively use existing codebases and design patterns to accelerate development. The conclusions highlight ChatGPT's potential in improving developer productivity, optimizing certain development tasks, and more efficiently allocating human resources in projects. However, the study also points out the need for human intervention to verify, correct, and improve generated code. The study contributes to understanding the practical usefulness of chatbots in real development scenarios and offers recommendations for integrating AI tools into the software development process.

KEYWORDS

Java, ChatGPT, automated code generation, production code, developer productivity

1 Introduction

In the modern information society, programming is one of the key competencies that ensure development and innovative progress. Java programming, due to its popularity and versatility, remains one of the most sought-after skills in software development (IEEE Spectrum, 2023). With the advent of machine learning and artificial intelligence, there is a reasonable need to automate the code generation process in order to increase efficiency and reduce development labor costs. In recent years, the integration of chatbots into software development has attracted considerable attention, with particular attention being paid to their potential in Java development. This study is aimed at exploring and defining areas of software development in Java, through chatbots, in particular ChatGPT. We explore scenarios in which a chatbot can effectively use existing code bases, libraries, and design patterns to accelerate development tasks in the Java ecosystem. Understanding this dynamic is crucial to understanding the practical usefulness of chatbots in real-world development scenarios.

The novelty of the study lies in the following: for the first time, a detailed classification of tasks for which the use of chatbots in Java development can be effective, as well as those that require qualified intervention by programmers, is proposed. The study examines specific

scenarios in which chatbots can use existing libraries and design patterns to speed up development processes, which highlights their practical usefulness.

The contribution of this article is as follows:

1. Conducting a systematic analysis of the effectiveness of code generation by the ChatGPT chatbot for typical Java development tasks.
2. Identifying specific areas and scenarios in the development of Java applications where the use of chatbots is most effective.
3. Identifying the limitations and potential risks of using chatbots in the Java development process.
4. Developing practical recommendations for integrating chatbots into the workflow of Java developers to improve productivity.
5. Assessing the impact of the use of chatbots on the allocation of human resources in software development projects.

Thus, this study aimed at exploring and defining areas of software development in Java, through chatbots, in particular ChatGPT. We explore scenarios in which a chatbot can effectively use existing code bases, libraries, and design patterns to accelerate development tasks in the Java ecosystem. Understanding this dynamic is crucial to understanding the practical usefulness of chatbots in real-world development scenarios.

The results of this study have important implications for Java development practice. Not only do they highlight the potential role of chatbots in increasing developer productivity and optimizing certain development tasks, but they also contribute to more efficient allocation of human resources in software development projects. Providing a detailed understanding of the practical utility of chatbots in Java development, this article considers software development to integrate chatbot technology more effectively.

This article explores this challenging area by combining two key spheres of computer science: software engineering and machine learning. In this study, we will analyze the efficiency of code generation by the ChatGPT chatbot for tasks usually solved within the framework of development in Java. In preparation for this study, a wide range of scientific papers and analytical materials over the past 2 years were analyzed.

Java is one of the most widely used and longest-running programming languages in the world (PYPL, 2024). Many industrial applications, enterprise systems, and banking programs use Java in their infrastructure (Free Educational Platform for Programmers FreeCodeCamp, 2024). Therefore, efficient and fast Java code generation is of great importance to ensure the reliability, performance, and security of such systems. Moreover, the use of the ChatGPT chatbot will reduce both the labor costs on the part of developers for a certain class of tasks, as well as the financial costs of companies for the staff of developers or budgets and time costs for the implementation of individual projects.

Previously, the scientific community had already considered the idea of optimizing code writing using ChatGPT. However, researchers have focused primarily on programming languages such as Python and JavaScript (Kashefi and Mukerji, 2023; Tian et al., 2023; Koubaa et al., 2023; Avila-Chauvet et al., 2023), (Feng et al., 2023), (Jayagopal et al., 2022). Therefore, efficient and fast Java code generation is of great importance to ensure the reliability, performance, and security of such systems. Moreover, the use of ChatGPT chatbot will allow for the reduction of both labor costs on the part of the developers for a certain class of tasks, as well as the financial costs of companies for the staff of developers or budgets and time costs.

Previously, the scientific community had already considered the idea of optimizing code writing using ChatGPT. However, researchers have focused primarily on programming languages such as Python and JavaScript (Zhao et al., 2024), especially after the release of ChatGPT in late November 2022. Large language models (LLM) have the ability to mimic human abilities in solving diverse and complex natural language processing and understanding tasks in various domains such as virtual assistants, chatbots, language translation, and sentiment analysis. In particular, ChatGPT, trained on a large and diverse dataset spanning multiple disciplines, demonstrates the ability to generate answers to a wide range of queries. The training data included many sources from fields such as science, literature, law, programming, finance, etc., totaling about 570 GB of data (Layton, 2023). The complex nature of the model, with over 175 million parameters, allows ChatGPT to generate answers to a wide range of queries with high efficiency.

A survey study by Feng et al. (2023) highlighted efforts by programmers to use ChatGPT for working with more than 10 languages. Python was the most frequently mentioned language in the majority of programming-related queries to ChatGPT, with the overall emotion of interaction with this LLM being overwhelmingly positive (see Figure 1). Figure 1 demonstrates the distribution of emotions expressed by users when interacting with ChatGPT for various programming languages. It is interesting to note that for Python, the most frequently mentioned language, the predominant emotion is anxiety, not fear, as was incorrectly stated earlier. This may indicate mixed feelings among users: on one hand, enthusiasm about AI capabilities in programming, on the other hand—concern about the potential impact on the programming profession.

Figure 1 shows the possibilities of generating code using ChatGPT based on Twitter and Reddit data. It was found that Python and JavaScript are the most frequently discussed programming languages, and ChatGPT is actively used for tasks such as debugging code, preparing for technical interviews, and completing academic assignments. Interestingly, people tend to feel more anxiety about the possibilities of code generation than joy, anger, or surprise.

The research also includes creating a dataset for rapid code generation that will be publicly available and evaluating the quality of the code generated using ChatGPT using Flake8. We hope that these results will contribute to improving software development and programming learning processes.

Summarizing the reviewed studies, with the evolution of development tools and technologies, researchers have repeatedly addressed the topic of optimizing code writing with code autocompletion tools (Biswas, 2023; Jayagopal et al., 2022; Vaithilingam et al., 2022). Many modern integrated development environments have built-in tools that allow you to use autocompletion to name variables, functions, classes, and comments. Such tools are able to analyze the context and substitute variables into function calls that match the type of the function based on the available variables in the current source code file. Examples are Blue-Pencil, Copilot, Flash Fill, Regae, and SnipPy. Research has shown that these tools do not always optimize code

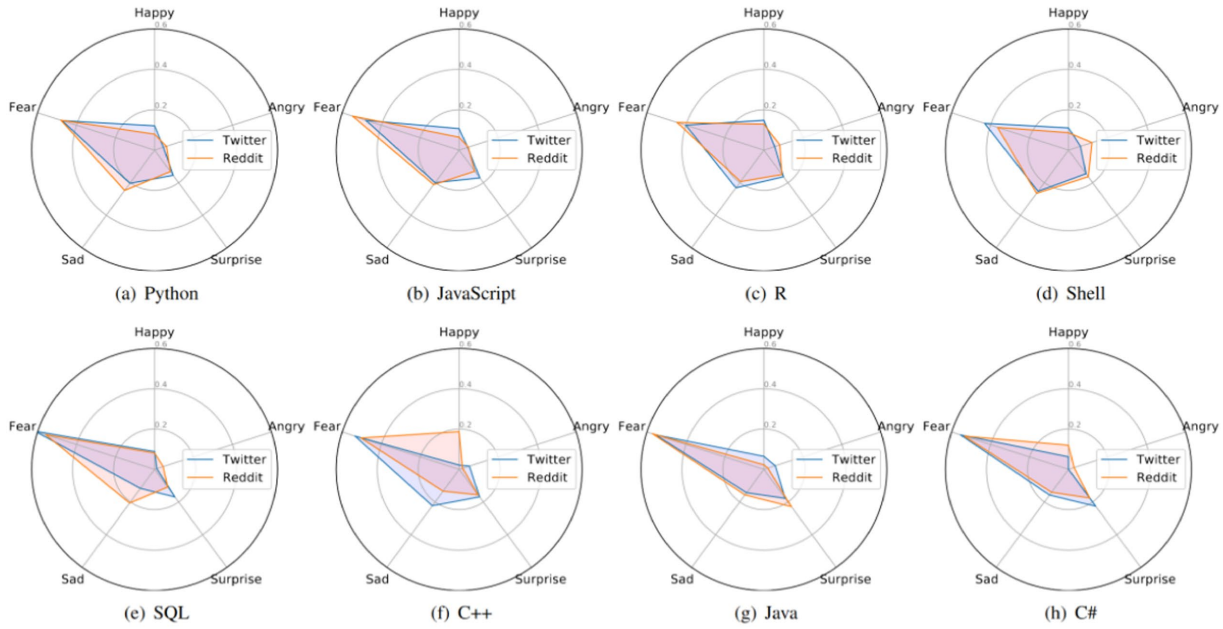

FIGURE 1
Expressed emotions from interaction with ChatGPT depending on the programming language (Feng et al., 2023).

efficiency and time spent writing code (Vaithilingam et al., 2022). Further research found (Barke et al., 2022), that the type of interaction with the code generator can be categorized into two global types:

1. Acceleration of code writing by auto-generation of code pieces, in case the programmer already has an idea of the algorithm to be used to solve the problem
2. Exploring possibilities of solving the problem when the developer does not have a clear solution plan

Within the framework of this article, the focus is on the first type of interaction—generating code pieces to perform clearly defined tasks. In this scenario, the programmer already has a predefined plan for how the architecture of the whole application will be constructed from these components. The primary goal of interacting with the code generation tool is to speed up the creation of building blocks of the program.

Since the majority of the research (Kashefi and Mukerji, 2023; Tian et al., 2023; Koubaa et al., 2023; Chauvet et al., 2023; Feng et al., 2023; Jayagopal et al., 2022) of code generation efficiency for the Java language were conducted on tools preceding the ChatGPT in November 2022, then it is reasonable to assume that such a complex language model, which analyzed huge libraries of open source, high-quality code, will be able to solve certain types of problems efficiently.

The article (Coello et al., 2024) «Effectiveness of ChatGPT in Coding: A Comparative Analysis of Popular Large Language Models» provides valuable data on the performance of various AI models in code generation tasks. The authors conducted a comprehensive analysis comparing ChatGPT with other popular language models. The study of this work allowed us to draw the following conclusions:

- ChatGPT has demonstrated high efficiency in solving a wide range of coding tasks.
- Certain areas have been identified where other models may be superior to ChatGPT, which indicates the need to choose an AI tool depending on the specific task.
- The study highlights the importance of continuous improvement and adaptation of AI models to the specific requirements of software development.

The analysis (Arefin et al., 2024) «Unmasking the Giant: A Comprehensive Evaluation of CHATGPT's Proficiency in Coding Algorithms and Data Structures», which provides a detailed analysis of ChatGPT's capabilities in key programming areas, allowed the authors of the work to come to the following results:

- A comparison of ChatGPT versions 3.5 and 4.0 showed significant progress in understanding and implementing complex algorithms and data structures.
- ChatGPT 4.0 demonstrates an improved ability to generate optimized code and offer effective solutions to algorithmic problems.
- Certain limitations have been identified in working with particularly complex or specific algorithms, which indicates the need for human supervision in critical tasks.

The study of the works of these authors allows us to:

- More accurately assess the current capabilities and limitations of ChatGPT in the context of Java development.
- Formulate more specific recommendations for using ChatGPT in various Java development scenarios.
- Propose strategies for integrating ChatGPT into the workflow, taking into account its strengths and potential limitations.

- Substantiate the need for a combined approach combining automated code generation with the expertise of a human developer.

Thus, the latest research from our predecessors significantly expands our understanding of the capabilities and limitations of AI in code generation. They confirm the potential of ChatGPT as a powerful support tool for Java developers while pointing out areas that require further improvement. These data allow us to more accurately define the role of AI in modern software development and propose effective strategies for its integration into Java development processes.

It is worth mentioning that the use of the previously mentioned code autocompletion tools may require the organization to share code with a third party (the tool vendor), which is often prohibited by organizations' regulations to maintain confidentiality and protect commercial intellectual property. The ChatGPT chatbot allows generating code not based on the context of the program file but in response to a user's request, which allows bypassing this restriction and complying with the organization's trade secret regulations.

In the following sections of the article, we will examine in detail the research methodology, present the results of experiments with ChatGPT in the context of Java development, analyze the obtained data, and discuss their significance for programming practice. We will conclude the article with findings and recommendations for integrating AI tools into the software development process.

2 Research methodology

In this study, we analyzed scientific and applied literature on the topic of interest. The empirical part presents the results of an experiment aimed at generating the code in response to a request and testing the performance of the resulting code. The outcome of our study is the synthesis of the results obtained and the generalization of ChatGPT-3.5's capabilities to generate code for solving industrial problems in Java.

For each task class, queries will be written in English according to ChatGPT's best query writing practices (Habr, 2023), after which the ChatGPT response will be analyzed for success or failure. If necessary, an expert analysis of the code quality will be conducted. In exceptional cases, the code will be adjusted to make it work.

In our study, we adhere to the following approaches for evaluating the generated codes:

1. Multi-level approach:

- Compilability: The basic level of verification.
- Functionality: Assessment of whether the code accomplishes the given task.
- Readability and maintainability: Analysis of the code's structure and style.

Adherence to Java best practices: Checking the code's compliance with generally accepted coding standards.

2. Qualitative analysis: In addition to automated checks, we conduct a qualitative analysis of the code based on expert evaluation by experienced Java developers.
3. Contextual relevance: We assess how well the generated code aligns with the task context and the requirements of modern Java development.

2.1 Statement of objectives

As part of this work, to analyze ChatGPT ability to generate code, the chatbot was asked to write code in Java to solve certain problems. Two types of problems were considered: simple and complex. Problems of both types arise in industrial development. Simple problems can be solved using one method in a class; complex ones require the interaction of several classes and methods. Table 1 shows both types of tasks.

Table 1 presents a classification of tasks used to test the code generation capabilities of ChatGPT.

Table 1 categorizes the tasks into two main types: simple tasks and complex tasks. These categories were used to evaluate how well ChatGPT performs in generating Java code for different levels of programming complexity. In this study, we define simple tasks as those that can be solved within a single method or class, do not require complex architecture, and do not involve interaction between multiple components. Complex tasks, on the contrary, include the development of multi-component systems, require an understanding of architectural patterns, and involve interaction between different parts of the application.

The simple tasks included operations like working with XML, JSON, multithreading, and data input/output. The complex tasks involved more intricate programming challenges such as writing MVC applications, building applications using Maven or Gradle, writing RESTful web services, and creating GUIs using various libraries.

This classification helps in understanding the strengths and limitations of ChatGPT in different areas of Java programming, ranging from basic operations to more advanced software architecture and design tasks.

In summary, Table 1 presents a classification of tasks used to test ChatGPT's code generation capabilities. The division into simple and complex tasks allows us to evaluate AI efficiency in various development scenarios, from basic operations to complex architectural solutions.

The following describes the tasks given to ChatGPT to solve. They are based on the practical requirements employers place on Java developers (Simplilearn, 2023).

TABLE 1 Tasks used to test ChatGPT's code generation ability.

Simple tasks	Complex tasks
Working with XML	Writing MVC (Model-View-Controller) Applications
Working with JSON	Building an application using Gradle
Using multithreading	Writing RESTful web services
Working with data entry	Writing JDBC (Java Database Connectivity)
Working with data output	Writing a GUI using the Swing Library
	Writing a GUI using the SWT Library
	Writing a GUI using the AWT Library

Certain classes of problems were not included for analysis as their solution required transferring the source commercial code to ChatGPT.

The following classes of problems were not included in this research:

- Code commenting
- Writing unit tests

The following snippets are meant to demonstrate efficient and performant Java code to meet production coding needs and optimize resource utilization.

2.2 Simple tasks

2.2.1 Working with XML

1. Using Java, parse an XML document into a Student object. The Student class has name, surname, age, and course. Write code that is production-ready, efficiently using system resources, and having maximum performance (XML to a Student object; Figure 2):
2. Using Java, serialize a Student object into an XML string. The Student class has name, surname, age, and course. Write a code that could be used in production, efficiently using system resources, and having maximum performance (serialization Student object in XML; Figure 3):

2.2.2 Working with JSON

3. Using Java, parse a JSON string into an object Student. The Student class has name, surname, age, and course. Write a code that could be used in production, efficiently using system resources, and having maximum performance (parse JSON string in Student object; Figure 4):
4. Using Java, serialize a Student object into a JSON string. The Student class has name, surname, age, and course. Write a code that could be used in production, efficiently using system resources, and having maximum performance (serialization Student object in JSON string; Figure 5):

2.3 Complex tasks

2.3.1 Writing MVC

Write a Java Spring MVC application to handle CRUD operations. The data object will be a Student, which has a name, surname, age, and course. Write a code that could be used in production, efficiently using system resources and having maximum performance.

2.3.2 Building an application using Maven

Write a Maven configuration to handle Spring MVC application. Write a code that could be used in production, efficiently using system resources and having maximum performance.

2.3.3 Building an application using Gradle

Write a Gradle configuration to handle Spring MVC application. Write a code that could be used in production, efficiently using system resources and having maximum performance.

2.3.4 Writing RESTful web services

Write a Java Spring RESTful application to handle CRUD operations. The data object will be a Student, which has a name, surname, age, and course. Write a code that could be used in production, efficiently using system resources and having maximum performance.

2.3.5 Writing JDBC (Java Database Connectivity)

Write a Java JDBC API implementation to retrieve and update records. The entity object will be a Student, which has a name, surname, age, and course. Write a code that could be used in production, efficiently using system resources and having maximum performance.

2.3.6 Writing a GUI using the Swing Library

Write a Java SWING application that can show a JPEG picture. Write a code that could be used in production, efficiently using system resources and having maximum performance.

2.3.7 Writing a GUI using the SWT Library

Write a Java SWT application that can show a JPEG picture. Write a code that could be used in production, efficiently using system resources and having maximum performance.

2.3.8 Writing a GUI using the AWT Library

Write a Java AWT application that can show a JPEG picture. Write a code that could be used in production, efficiently using system resources and having maximum performance.

3 Results

This section presents the results obtained during interaction with ChatGPT-3.5 with a description of the launch result obtained from the ChatGPT code and brief comments on this code and interaction with it.

3.1 Extension of results to simple problems

In each category of simple tasks (working with XML, JSON, multithreading, and data input and output), several subtasks were tested. The numbering in each category corresponds to these subtasks. For example, in the 'Working with XML' category, points 1–5 refer to various XML operations, such as deserialization, serialization, element search, etc.

3.1.1 Working with XML

1. The code is working. The XML was deserialized into an object. The chatbot did not indicate which dependencies needed to be imported but provided an example to test the functionality of the generated code.
2. The code is working. The serializer provided by the chatbot was missing one import.
3. The code is working. The chatbot indicated where to insert the search file and the search key.
4. The code is working. The chatbot indicated where to insert the name of the element for which we are searching the minimum and maximum values.

```java
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;
import java.io.File;

public class StudentXMLParser {
    public Student parseStudentXML(String filePath) throws JAXBException {
        File file = new File(filePath);
        JAXBContext jaxbContext = JAXBContext.newInstance(Student.class);
        Unmarshaller unmarshaller = jaxbContext.createUnmarshaller();

        return (Student) unmarshaller.unmarshal(file);
    }
}
```

FIGURE 2
XML processing for the student object.

5. The code is working. The chatbot also provided an example on which you can immediately check the functionality of the code.

3.1.2 Working with JSON

1. The code is working. Parsing a string into a Java object—the chatbot did not write getters and setters, but immediately wrote an example of a JSON string on which it is possible to test the parsing result.
2. The code is working. Serialization of Java object to JSON was successful. The chatbot did not write getters and setters.
3. The code is working. Searching for a value by key in a JSON string was successful. The chatbot also provided an example on which you can immediately check the functionality of the code.
4. The code is working. Searching for the minimum and maximum numbers in the JSON array worked correctly. The chatbot also provided an example on which you can immediately check the functionality of the code.
5. The code is not working. The code issued by the chatbot compiled but did not work.

3.1.3 Using multithreading

1. The code is working. However, the test case that accompanied the code did not check the operation of multithreading.
2. The code is working. Implementation of the code received from the chatbot involves the creation of only one class object and is protected from errors in a multi-threaded environment.
3. The code is working. The chatbot used API Java 8 for parallel stream processing.
4. The code is not working. The code compiled but ended up in an infinite loop.
5. The code is working. ChatGPT generated the code using standard APIs Java for multi-threaded parallel processing of multiple tasks. The code was provided with clear explanatory comments and supplemented with a test script to check its functionality.

3.1.4 Working with data entry

1. The code is working.
2. The code is not working. The program entered an infinite loop and would never finish.
3. The code is working. The chatbot provided an example to test the generated code.
4. The code is working. The chatbot provided an example to test the generated code.
5. The code is working. The chatbot suggested where to insert the file for reading in the method.

3.1.5 Working with data output

1. The code is working.
2. The code is working. In addition to creating a function, the chatbot also showed how standard output works in Java.

```java
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import java.io.StringWriter;

public class StudentXMLSerializer {
    public String serializeStudentToXML(Student student) throws JAXBException {
        JAXBContext jaxbContext = JAXBContext.newInstance(Student.class);
        Marshaller marshaller = jaxbContext.createMarshaller();

        // Настройка для форматирования вывода
        marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);

        StringWriter writer = new StringWriter();
        marshaller.marshal(student, writer);

        return writer.toString();
    }
}
```

FIGURE 3
Serialization of the student object in XML.

```java
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.JsonMappingException;
import java.io.IOException;

public class StudentJsonParser {
    private final ObjectMapper objectMapper;

    public StudentJsonParser() {
        this.objectMapper = new ObjectMapper();
    }

    public Student parseJsonToStudent(String jsonString) throws IOException {
        return objectMapper.readValue(jsonString, Student.class);
    }
}
```

FIGURE 4
Developing a JSON string into a student object.

3. The code is working. The chatbot provided an example of checking the functionality of the generated code.
4. The code is working. The chatbot provided an example of checking the functionality of the generated code.
5. The code is working. The chatbot provided an example of checking the functionality of the generated code.

3.2 Extension of results to complex problems

3.2.1 Writing MVC

The application written by the chatbot did not compile. The Student class lacked getters and setters. The StudentRepository class

```java
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.core.JsonProcessingException;

public class StudentJsonSerializer {
    private final ObjectMapper objectMapper;

    public StudentJsonSerializer() {
        this.objectMapper = new ObjectMapper();
    }

    public String serializeStudentToJson(Student student) throws JsonProcessingException {
        return objectMapper.writeValueAsString(student);
    }
}
```

FIGURE 5
Serialization the student object in JSON string.

was missing the @Repository annotation. The StudentController controller was created as a REST controller, not an MVC controller to return HTML pages. Even after a couple of iterations with errors noted, the application did not build.

3.2.2 Building the application using Maven

After inserting the created configuration, the application did not build due to a lack of libraries.

3.2.3 Building an application using Gradle

The created configuration contained an error.

Thus, the application could not be built using the configuration written by the chatbot.

3.2.4 Writing RESTful web services

The code created by the chatbot again contained flaws. The Student class lacked getters and setters. The StudentRepository class was missing the @Repository annotation. The controller worked directly with the StudentRepository class; no service was created to work with the Repository. Thus, the single responsibility principle of the SOLID paradigm was violated. Moreover, the controller did not have enough prescribed path mappings, so despite the fact that the application started, it was mostly unusable—it was impossible to get a list of all entities or add an entity.

3.2.5 Writing JDBC (Java Database Connectivity)

The code is not working. The application did not compile as possible errors were not handled when connecting to the database.

3.2.6 Writing a GUI using the Swing Library

After the code provided by the ChatGPT chatbot was copied and the path to the required image was added, the application compiled and was successfully launched. One of the shortcomings was that it contained one extra library import. The entire application was one class with the main function included in it.

3.2.7 Writing a GUI using the SWT Library

The ChatGPT chatbot wrote the code directly in the main function, but after adding the path to the image file, the code ran successfully and a window with the expected image was displayed. The entire application was one class with the main function included in it.

3.2.8 Writing a GUI using the AWT Library

The generated code was missing a closing parenthesis. However, after adding the missing bracket and specifying the path to the image file, the code compiled and ran. The entire application was one class with the main function included in it.

4 Discussion

An overall analysis of the results for simple and complex problems is given in Tables 2, 3, respectively. Table 2 shows the number of successful cases and the total number of proposed tasks, broken down by topic. Table 3 shows the results of running ChatGPT-3.5 complex tasks, noting whether the result of running the code was successful or not.

4.1 Simple tasks

These studies examine in detail how effectively ChatGPT copes with tasks of various levels of complexity—from the simplest to the most complex. In studies such as (Brown et al., 2020), by the authors of GPT-3, the ability of the model to solve both simple and complex problems, including arithmetic, logic, programming, text comprehension, and code generation, is discussed. Some studies, such as (Chen and et al., 2021) (Codex), analyze the performance of the model in solving programming problems of various levels of complexity—from simple loops to complex algorithms. In particular, the ability of models to debug and generate code is evaluated. Studies

TABLE 2 Summary analysis of the results based on the simple task execution.

Task name	Number of solved tasks	Total number of tasks
Working with XML	5	5
Working with JSON	4	5
Using multithreading	4	5
Working with data entry	4	5
Working with data output	5	5

TABLE 3 Summary analysis of the results based on completing complex tasks.

Task name	Positive result	Negative result
Writing MVC		✓
Building an application using Maven		✓
Building an application using Gradle		✓
Writing RESTful web services		✓
Writing JDBC (Java Database Connectivity)		✓
Writing a GUI using the Swing Library	✓	
Writing a GUI using the SWT Library	✓	
Writing a GUI using the AWT Library	✓	

such as (Bubeck et al., 2023) examine the ability of models to cope with tasks comparable to those found in cognitive tests for logical thinking, including both simple and complex problems. Works, for example (OPEN AI, 2023), often include testing ChatGPT on simple and complex natural language processing tasks—from understanding context to creating complex texts, which allows you to identify how the model copes with various levels of complexity.

In addition, for clarity, a diagram (Figure 6) is provided to show the number of successfully completed simple tasks.

Based on the table above, we can conclude that when working with relatively simple problems that can most often be solved within one method, ChatGPT is quite helpful when working with XML, JSON, multi-threaded processing, and data input/output.

Thus, in production development, programmers can write short queries to obtain work functions. However, errors in the generated code leave the programmers responsible for checking and validating the quality of the code.

The distribution table shows that in the majority of cases, ChatGPT cannot cope with complex tasks. Other researchers made similar conclusions in their studies (Barke et al., 2022; Ferdowsifard et al., 2020), pointing out that the more difficult a task we assign to LLM, the more its response degrades.

Based on the table with the results of complex tasks, we can conclude that ChatGPT can easily write user interfaces for desktop applications. However, it is important to note that the tasks involving writing user interfaces were relatively simple. The chatbot was only required to display a picture in the window. Thus, we believe that the general findings of this study are also valid for the tasks related to writing a user interface—ChatGPT can write simple methods and classes for user interfaces in response to programmer requests. The programmer should build a full-fledged application with transitions between windows and blocks using, should they wish to do so, ready-made templates and developments from ChatGPT.

Separately, it is worth mentioning the task of building applications into executable archives using the Gradle and Maven build systems. With other complex tasks, the answer issued by ChatGPT can be iteratively brought to a workable state by adding and changing the code (however, correcting and modifying the generated code looks like a lengthy and irrational task—writing a solution from scratch is faster). However, ChatGPT was completely unable to build and specify the necessary libraries, so it is more practical for programmers to use hints from integrated development environments, which may offer to add the necessary libraries to the project by analyzing the semantics of the classes and methods used.

Thus, we categorized Java development subject areas that ChatGPT-3.5 solves successfully and unsuccessfully. The study found that Java code for simple tasks that require interaction with JSON, XML, and multi-threaded input and output data can be successfully generated by ChatGPT-3.5.

However, the code generated by ChatGPT-3.5 for solving complex problems often does not work as expected. For complex problems, a reasonable approach would be to split complex problems into simple ones and generate code for them. Based on the blocks of code that solve simple problems, a complete program can be developed that solves a non-trivial problem.

Furthermore, it is worth noting that one example of modern design patterns that can be used to compare generated code is the AdapT pattern, developed for implementing smart contracts that process transactions of congruous types. This pattern, described in the works "AdapT: A reusable package for implementation smart contract that processing transaction of congruous types" (Górski, 2024a) and "Smart Contract Design Pattern for Processing Logically Coherent Transaction Types," provides a reference implementation for comparison (Górski, 2024b).

When comparing the generated code with the AdapT pattern and other established design patterns, we propose the following evaluation criteria:

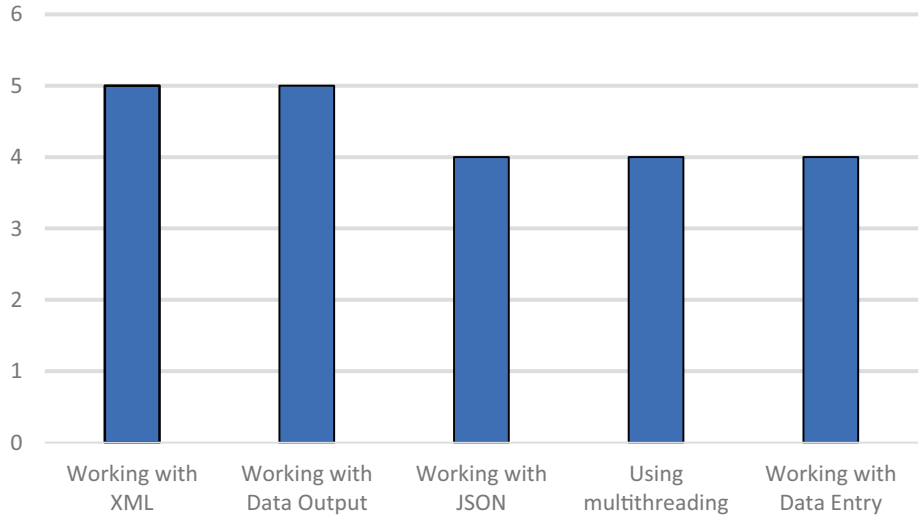

FIGURE 6
Chart of successfully completed simple tasks by category.

1. Structural conformity: How well does the structure of the generated code align with the principles of the design pattern?
2. Functional equivalence: Does the generated code perform the same functions as the reference implementation?
3. Efficiency: How does the generated code compare to the reference implementation in terms of performance and resource utilization?
4. Extensibility and maintainability: How easy is it to modify and extend the generated code compared to the reference implementation?
5. Adherence to SOLID principles and other best practices: Does the generated code comply with the fundamental principles of object-oriented design?

To conduct such an analysis, we propose the following methodology:

- Selection of representative tasks that can be solved using the AdapT pattern and similar patterns.
- Code generation using ChatGPT to solve these tasks.
- Comparison of the generated code with reference implementations based on the above criteria.
- Quantitative and qualitative evaluation of results, involving experts in Java development.

This approach allows for a more objective assessment of the quality of generated code and helps identify areas where AI code generation can be most effective, as well as reveal limitations and potential problems.

In future research, we plan to conduct a detailed analysis based on this methodology, which will allow us to provide more specific recommendations on the use of AI code generation in real projects and determine the optimal scenarios for integrating such tools into the software development process.

This addition to the Discussion section addresses the issues raised and offers a specific plan to improve the quality assessment of the generated code. We are grateful for the recommendations and links to articles regarding AdapT, which will undoubtedly enrich our research and help in the further development of this topic.

5 Conclusion based on the results

The experimental results highlight both the potential and limitations of using ChatGPT-3.5 for various programming tasks. While AI produced functional code for some simple problems, it encountered more complex problems, often resulting in non-functional or incomplete solutions. The need for human intervention and review to correct and improve the generated code is obvious, highlighting the importance of human involvement in the software development process. We also classified the tasks for the effective use of industrial code and developed recommendations for the use of chatbot assistance. In the modern information society, programming is a key competency that ensures development and fosters innovative progress. It also helps outline the boundaries in which human intervention remains indispensable, that is, areas where manual programming by developers is still required.

In addition to suggesting a classification of the tasks for which chatbots can be of help and the ones that are more effectively solved by human programmers, we also considered scenarios where a chatbot can effectively leverage existing codebases, libraries, and design patterns to speed up development tasks in the Java ecosystem. This is critical to understanding the practical utility of a chatbot in real-world development scenarios. This article provides guidance on tasks that cannot be entrusted to ChatGPT and require the involvement of skilled programmers. By describing scenarios where manual programming is preferable, the article offers pragmatic advice to developers, thereby improving their decision-making process when using chatbot support.

The findings of the article have important implications for Java development practice because they shed light on the potential role of chatbots in increasing developer productivity, optimizing certain development tasks, and enabling more efficient allocation of human resources in software projects.

Future directions of research may include expanding the cluster of successful simple tasks that ChatGPT can handle, as well as analyzing and clustering hints that generate the most optimal and efficient code.

6 Threats to validity and study limitations

While our study provides insights into AI-assisted Java programming, it is important to acknowledge certain limitations:

Scope: Our research focuses on ChatGPT-3.5, which may not fully represent the capabilities of other AI models or future versions.

Evaluation: Despite our efforts to use objective criteria, there is an inherent element of subjectivity in assessing code quality.

Task Range: The selected tasks, while diverse, may not encompass all possible scenarios in Java development.

Task Classification: Our definitions of 'simple' and 'complex' tasks may not universally apply to all software development contexts.

Despite these limitations, this study contributes to the understanding of AI's current role in Java programming. It offers a snapshot of AI capabilities, potentially guiding future research and practical applications in software development. While acknowledging these constraints, we believe our findings provide valuable insights for both researchers and practitioners in the field.

7 Conclusion

The experimental results demonstrate both the potential and limitations of using ChatGPT-3.5 to perform various programming tasks. Although artificial intelligence can successfully generate functional code for simple tasks, when working with more complex tasks, it often encounters problems that lead to the creation of non-functional or incomplete solutions. This highlights the need for human intervention to check, correct, and improve the code, which in turn points to the importance of human involvement in the software development process.

We have classified tasks according to their effectiveness when using automated development tools and created recommendations for the use of chatbots. The boundaries were also outlined where human participation remains critically important, that is, where manual programming by professional developers is required. In addition, a number of scenarios were considered in which chatbots can effectively use existing code libraries and design patterns to speed up the development process in the Java ecosystem. This is an important aspect that illustrates the practical usefulness of chatbots in real conditions. The proposed recommendations and conclusions are of practical importance for Java development, as they help developers make more informed decisions about when to use chatbots and when human intervention is necessary.

This study aimed to explore the potential and limitations of using ChatGPT-3.5 in the context of Java software development. Based on our experiments and analysis, we have reached the following conclusions:

1. **ChatGPT's Potential and Limitations**: The research demonstrated that ChatGPT-3.5 is capable of generating functional code for simple programming tasks. However, when solving more complex problems, AI often creates non-functional or incomplete solutions, highlighting the need for human intervention.
2. **Task Classification**: We successfully developed a classification of tasks, identifying areas where chatbot assistance can be most effective and those where human developer involvement remains indispensable. This classification serves as a practical guide for optimizing the development process.
3. **Integration with Existing Resources**: The study revealed scenarios in which a chatbot can effectively utilize existing codebases, libraries, and design patterns, potentially accelerating the development process in the Java ecosystem.
4. **Role of Human Developers**: The results confirm the critical importance of human participation in the development process. We provided recommendations for tasks requiring programmer expertise and described scenarios where manual programming is preferable.
5. **Practical Significance**: Our findings have direct application in Java development practices, offering ways to increase developer productivity and optimize resource allocation in software projects.
6. **Methodological Improvements**: During the study, we identified the need to develop more formalized criteria for evaluating the quality of generated code, including comparison with existing design patterns such as AdapT.
7. **Prospects for Further Research**: Future studies may focus on expanding the cluster of tasks successfully solved by ChatGPT, analyzing and optimizing prompts for generating the most effective code, and developing standardized methods for evaluating AI-generated code.

In conclusion, our research demonstrates that while ChatGPT has significant potential in software development, its use should be carefully integrated into existing development processes. The optimal application of AI in programming requires a balance between automation and human expertise, opening new opportunities for improving the efficiency and quality of software development.

As part of the project IRN AP19678174 on the topic "Development the theory and methodology for formation development programs of the RK during the transformation the economy into an innovative".

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

MU: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Resources, Software, Writing – original draft, Writing – review & editing. LB: Conceptualization, Investigation, Methodology, Project administration, Software, Supervision, Writing – original draft. GM: Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – review & editing. AE:

Conceptualization, Data curation, Formal analysis, Methodology, Resources, Visualization, Writing – original draft. TM: Formal analysis, Investigation, Resources, Software, Validation, Writing – review & editing.

Funding

The author(s) declare that no financial support was received for the research, authorship, and/or publication of this article.

Acknowledgments

This article was produced as part of the project IRN AP19678174 on the topic "Development of Theory and Methodology for Kazakhstan's Development Programs during Economic Transformation to Innovation. ChatGPT-3.5 was used on various Java development tasks as well as for the methodology of the manuscript and the analysis of scientific literature and empirical testing.

Conflict of interest

Author AY were employed by Kazmetengineering LLP, Almaty, Kazakhstan.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

Arefin, S. E., Heya, T. A., Al-Qudah, H., Ineza, Y., and Serwadda, A. (2024). Proceedings of the 16th International Conference on Agents and Artificial Intelligence. Unmasking the Giant: A comprehensive evaluation of ChatGPT's proficiency in coding algorithms and data structures. Texas Tech University, Lubbock. 412–419.

Avila-Chauvet, L., Mejía, D., and Acosta Quiroz, C. O. (2023). Chatgpt as a support tool for online Behavioral task programming. Available at SSRN: https://ssrn.com/abstract=4329020 (Accessed January 18, 2023).

Barke, S., James, M. B., and Polikarpova, N. Grounded Copilot: How programmers interact with code-generating models. (2022). arXiv:2206.15000v3 [cs.HC]. doi: 10.48550/arXiv.2206.15000

Biswas, S. (2023). Role of ChatGPT in computer programming. *Mesopot. J. Comp. Sci.*, 9–15. doi: 10.58496/MJCSC/2023/002

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., et al. (2020). "34th conference on neural information processing systems" in Language models are few-shot learners. arXiv:2005.14165v4 [cs.CL]. doi: 10.48550/arXiv.2005.14165

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., et al. Sparks of artificial general intelligence: Early experiments with GPT-4. (2023). arXiv:2303.12712v5 [cs.CL]. doi: 10.48550/arXiv.2303.12712

Chauvet, L., Cruz, D., and Quiroz, C. Chatgpt as a support tool for online Behavioral task programming. SSRN. (2023). Available at: https://ssrn.com/abstract=4329020 (Accessed January 18, 2023).

Chen, M., Tworek, J., Jun, H., Yuan, Q., Oliveira Pinto, H. P., Kaplan, J., et al. (2021). Evaluating large language models trained on code. arXiv:2107.03374. doi: 10.48550/arXiv.2107.03374

Coello, C., Alimam, M., and Kouatly, R. Effectiveness of ChatGPT in coding: a comparative analysis of popular large language models. Digital 1, (2024). 114–125. doi: 10.3390/digital4010005

Feng, Y., Vanam, S., Cherukupally, M., Zheng, W., Qiu, M., and Chen, H. (2023). "47th IEEE annual computers, software, and applications conference, COMPSAC 2023" in Investigating code generation performance of ChatGPT with crowdsourcing social data. (Torino, Italy: IEEE). 26–30.

Ferdowsifard, K., Ordookhanians, A., Peleg, H., Lerner, S., Polikarpova, N. (2020). UIST'20. small-step live programming by example. 614–626. doi: 10.1145/3379337.3415869

Free Educational Platform for Programmers FreeCodeCamp. (2024). What is Java used for? The Java programming language and Java platform. Available at: https://www.freecodecamp.org/news/what-is-java-used-for/ (Accessed April 25, 2023).

Górski, T. (2024a). AdapT: a reusable package for implementing smart contracts that process transactions of congruous types. *Software Impacts*. 21:100694. doi: 10.1016/j.simpa.2024.100694

Górski, T. (2024b). Smart contract design pattern for processing logically coherent. *Appl. Sci.* 14:2224. doi: 10.3390/app14062224

Guo, B., Zhang, X., Wang, Z., Jiang, M., Nie, J., Ding, Y., et al. How close is ChatGPT to human experts? Comparison Corpus, evaluation, and detection. (2023). arXiv:2301.07597v1 [cs.CL]. doi: 10.48550/arXiv.2301.07597

Habr. (2023). Experience working with ChatGPT using the example of writing Java code to solve a typical problem and some conclusions and reasoning. Available at: https://habr.com/ru/articles/744368/ (Accessed June 28, 2023).

Hu, Y., Ahmed, U. Z., Mechtaev, S., Leong, B., and Roychoudhury, A. (2019). ACM international conference on automated software engineering.» Re-factoring based program repair applied to programming assignments. *Proceedings of the 34th IEEE*. San Diego, California: ASE '19. ASE '19. 388–398.

IEEE Spectrum. (2023). The top programming languages 2024 typescript and rust are among the rising stars//IEEE Spectrum. Available at: https://spectrum.ieee.org/the-top-programming-languages-2023 (Accessed March 19, 2023).

Jayagopal, D., Lubin, J., and Chasins, S. (2022). In proceedings of the 35th annual ACM symposium on user Interface software and technology (UIST '22). New York: Exploring the Learnability of Program Synthesizers by Novice Programmers. Association for Computing Machinery.

Kashefi, A., and Mukerji, T. ChatGPT for programming numerical methods. (2023). doi: 10.48550/arXiv.2303.12093

Khabisi, M., Roudini, G., Barahuie, F., and Sheybani, H. (2023). Evaluation of phase change material-graphene nanocomposite for thermal regulation enhancement in buildings. 9:e21699. doi: 10.1016/j.heliyon.2023.e21699

Koubaa, A., Qureshi, B., Ammar, A., Khan, Z., Boulila, W., and Ghouti, L. (2023). Humans are still better than ChatGPT: case of the IEEEXtreme competition. *Heliyon* 9:e21624. doi: 10.1016/j.heliyon.2023.e21624

Layton, D. ChatGPT—Show me the data sources. (2023). Available at: https://medium.com/@dlaytonj2/chatgpt-show-me-the-data-sources-11e9433d57e8 (Accessed May 15, 2024).

Liu, H., Ning, R., Teng, Z., Liu, J., Zhou, Q., and Zhang, Y. Evaluating the logical reasoning ability of ChatGPT and GPT-4. (2023). doi: 10.48550/arXiv.2304.03439

OPEN AI. (2023). Available at: https://www.thinkhousehq.com/the-youth-lab/2023-the-year-of-open-ai (Accessed May 15, 2024).

PYPL. (2024). Index (PopularitY of Programming Languages). Available at: https://pypl.github.io/PYPL.html (Accessed January 14, 2024).

Qin, C., Zhang, A., Zhang, Z., Chen, J., Yasunaga, M., Ding, Y. (2023). ChatGPT a general-purpose natural language processing task solver? In Proceedings of the 2023 conference on empirical methods in natural language processing. (Singapore: Association for Computational Linguistics), 1339–1384.

Simplilearn. Java Developer Job Description: Role, Responsibilities, and More. (2023). Available at: https://www.simplilearn.com/java-developer-job-description-article (Accessed August 7, 2023).

Tian, H, Lu, W., Li, T. O., Tang, X., Cheung, S-C., Klein, J., et al. Is ChatGPT the ultimate programming assistant - how far is it?. Vol. 1. (2023). doi: 10.48550/arXiv.2304.11938

Vaithilingam, P., Zhang, T., and Glassman, E. CHI EA '22: Extended abstracts of the 2022 CHI conference on human factors in computing systems.» expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. New Orleans, LA, USA: Association for Computing Machinery, (2022). 1–7.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., et al. A survey of large language models. (2024). doi: 10.48550/arXiv.2303.18223