UDC 004.8:004.42

2. ARTIFICIAL INTELLIGENCE IN SOFTWARE ENGINEERING

Salnikov D.A., Master's degree student, group 326401

Belarusian State University of Informatics and Radioelectronics[1], Minsk, Republic of Belarus

Subotkina I.G. – Assistant professor

Annotation. Research has been conducted in the field of modern methods and tools of artificial intelligence and neural networks, to investigate how software engineers can facilitate their development. It has been established what the current state of the software development industry is, and how inexperienced developers can cope with the tasks assigned to them.

Keywords. software engineers, artificial intelligence, development, NLP – natural language processing.

The software development job market is currently undergoing major changes. Employers are less interested in hiring inexperienced developers, which in turn leads to inflated demands for young specialists who have just begun their journey into computer science. The trend is observed both in the Western market and in the markets of Belarus and Russia [1].

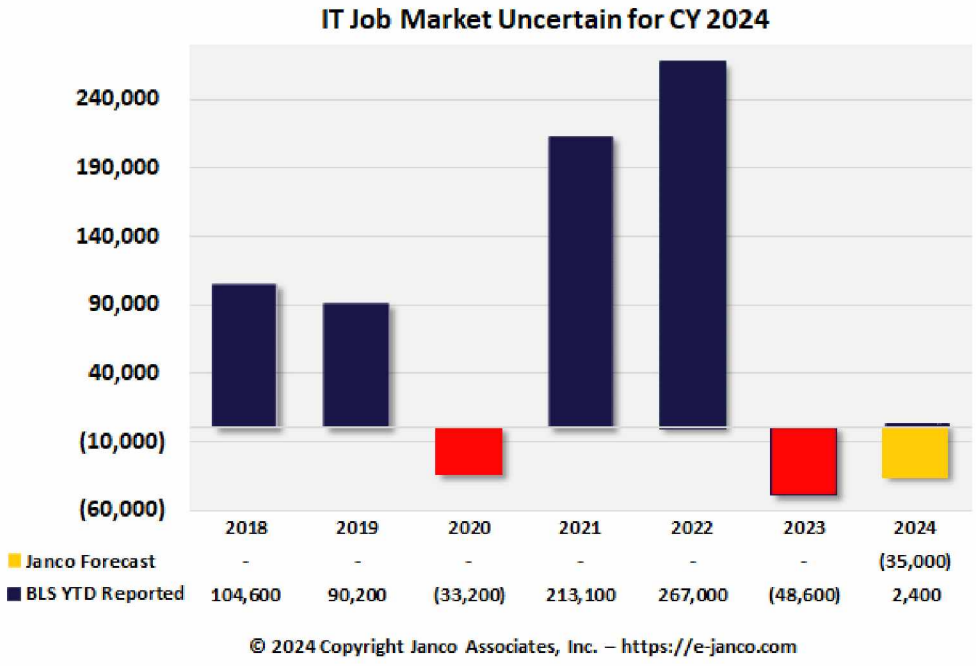

Figure 1 – IT job market 2024

At the same time, artificial intelligence technologies are developing daily, and literally every month trained models capable of generating program code appear in the public domain completely free of charge. This information leads the average person to certain thoughts. Are modern neural networks capable of writing code at least at the level of a novice developer, or even better?

At the moment, the smartest neural network models can carry out important tasks in software development. Next, use cases for AI in software development will be presented.

Prototyping. Effective time management is of utmost importance during the prototyping phase of any software development project. This crucial step involves creating preliminary versions of the software to test and verify its functionality. By using automated code generation using artificial intelligence, developers can speed up the prototyping process, enabling rapid iterations and refinements. With this capability, developers can use AI to quickly explore different design possibilities, identify potential problems early, and improve collaboration between teams. Consequently, the use of AI in prototyping speeds up development cycles, promotes innovation, and ultimately leads to robust and user-centric software solutions.

Debugging. AI-powered error detection uses advanced algorithms to identify software problems by analyzing errors in pattern code, data flow, and logs. Designed using sophisticated algorithms, it can also scrutinize code patterns, data streams, and error logs to quickly and accurately identify anomalies and anomalies that may indicate errors. This containment approach allows developers to find and avoid bugs early,

reducing their impact on on-time development. By quickly scanning a large database of code and studying historical error data, AI algorithms make it easier to identify errors and improve the overall quality of software without causing problems before reaching end users.

Explanation of written program code. AI-powered code explanation systems use natural language processing (NLP) and machine learning techniques to analyze code and create human-readable explanations. These explanations provide insight into the purpose, functionality, and logic of code snippets, making it easier for developers to understand and work with unfamiliar or complex code. Benefits of AI code explanation include: simplify complex code bases to speed up onboarding for new developers, improve collaboration among software development teams and knowledge sharing by standardizing code discussions, tailor code explanations with AI's adaptive learning capabilities and ability to process developer comments to understand the specific needs and tastes of the development team.

Refactoring. Code refactoring helps one analyze code and identify areas of improvement in a software development project. With AI-powered code refactoring systems that use machine learning algorithms and pattern recognition techniques, you can access more direct suggestions for refactoring strategies such as restructuring code, renaming variables, removing duplicate code, or applying design patterns to improve quality code and ease of maintenance. Some benefits of AI-based code refactoring include the following:

- AI algorithms can detect code smells and anti-patterns that can lead to bugs, performance issues, and codebase maintenance issues. Detection algorithms allow developers to proactively address these flaws, resulting in simpler maintenance code;
- AI-powered code refactoring solutions improve code readability, modularity, and efficiency. By leveraging industry best practices and coding norms, systems can suggest refactoring actions to help developers meet coding standards and improve code quality;
- AI algorithms can study code repositories and software development practices to observe changes and suggest refactoring based on this information.

Predictive analysis. AI-powered predictive analytics tools use historical project data, machine learning algorithms, and statistical models to predict future project outcomes. These algorithms can accurately estimate project timelines, resource requirements, and potential risks by analyzing patterns, trends, and dependencies. The benefits of using AI for predictive analytics in project management include the following:

- AI algorithms can help determine realistic project goals and estimates, helping to ensure project completion, resource allocation, and bottleneck reduction;
- AI-powered predictive analytics can analyze obstacles and detect hazards and bottlenecks early by analyzing past project data, team performance, external dependencies, and market conditions. This helps project managers reduce risks, optimize resource allocation and meet deadlines;
- artificial intelligence systems can adapt to project developments and make predictions and recommendations in real time.

This agile approach to project management allows teams to quick respond to changing demands and changing circumstances, promoting adaptability and flexibility.

Automatic code testing. Using AI-assisted automated testing techniques help one analyze code, identify potential vulnerabilities, and simulate various test scenarios. By automating the testing process, AI algorithms can execute test cases faster and more efficiently than manual testing methods, saving time and resources. The benefits of using AI for automated testing and quality assurance include the following:

- artificial intelligence algorithms can run more test scripts to test the functionality of the software. This feature reduces the risk of critical bugs escaping testing and reaching consumers;
- artificial intelligence systems can recognize complex software behavior patterns. They can also identify bugs, performance bottlenecks, and security issues that manual testing might miss. This improves the quality of the software and reduces post-deployment hassles;
- developers can quickly find and fix defects using artificial intelligence algorithms that analyze test results and record information. This reduces debugging time, allowing developers to meet deadlines and produce high-quality software.

Documentation. Developers can work with intelligent, AI-powered documentation flows to analyze code, comments, and resource assets. These systems can automatically recall relevant information, include descriptive documentation, and provide contextual explanations for code snippets, functions, and modules. The benefits of creating documentation using artificial intelligence include the following points:

- automates tedious technical documentation for developers;
- AI documentation systems analyze code repositories and track code versions to automatically update relevant documentation pages to reflect the current code base;
- artificial intelligence algorithms improve the readability of documentation. They can simplify technical jargon, explain complex concepts, and provide detailed explanations, making documents more accessible to technical and non-technical audiences.

Copilot tools. Copilot AI tools such as GitHub Copilot, Codeium, and Whisperer are designed to help developers write scripts using machine learning techniques.

GitHub Copilot: GitHub Copilot is an artificial intelligence-powered code completion tool developed by GitHub in collaboration with OpenAI. It uses GPT-3.5, the same underlying technology as ChatGPT, to generate code suggestions as developers write in their preferred code editor. Copilot analyzes context, including existing code, comments, and function signatures, to provide relevant code additions, snippets, and even entire functions. It aims at saving time and increasing productivity by automating repetitive tasks and helping with code generation.

Codeium: Codeium is another AI-based tool that helps in writing code scripts. While specific details about Codeium are not available in training data as of September 2021, similar tools often use machine learning algorithms to analyze code patterns, learn from existing code bases, and generate code snippets or entire functions based on context and developer intent.

Whisperer: Whisperer is an artificial intelligence tool developed by OpenAI that helps translate code from natural language. It aims to bridge the gap between domain experts and developers by allowing users to describe their intent in plain English or other natural languages, and then generate code based on that description. Whisperer uses machine learning techniques to understand the user's intent and generate code that meets their requirements [2].

It is important to note that interaction with artificial intelligence tools to create software is used in natural language. Of course, writing the right prompts for the model plays an important role here. The more understandable the request for the model is from a technical point of view, the higher the final result provided by the artificial intelligence service will be.

Jensen Huang, CEO of Nvidia, recently said that fast-growing artificial intelligence is becoming a competitor for developers. The 61-year-old explained that learning coding was once an all-important task, but in today's world, it holds little value. “Over the last 10-15 years, almost everybody who sits on a stage like this would tell you that it is vital that your children learn computer science, everybody should learn how to program. In fact, it is almost exactly the opposite,” he said. J. Huang also said that it is necessary to develop technologies in the direction of natural language for artificial intelligence understand user requests better. “It is our job to create computing technology such that nobody has to program and that the programming language is human. Everybody in the world is now a programmer. This is the miracle of AI,” he said. J. Huang believes that children should not be forced to learn programming, but rather focus on their individual skills [3].

In modern countries, you can increasingly see vacancies with the title “Prompt engineer”. This is a specialist who works with natural language models of artificial intelligence in order to improve their understanding and interaction with the requests of the average user. AI prompt engineers serve as intermediaries between machine learning models and the humans who query them. The job of an AI prompt engineer is to develop a set of inputs and train the models to produce the best and desired outputs back to the user. As such, the role involves writing text-based prompts and feeding them into the back end of AI tools to enable them to perform tasks, such as writing an essay, generating a blog post or creating a sales email with the proper tone and information. Because AI systems lack intuition, they're dependent on human input to understand human language and questions to produce effective prompts. Well-crafted prompts play a pivotal role in enabling the AI model to grasp the user's intention and context, ultimately resulting in responses that are both accurate and pertinent. To successfully build and optimize prompts for AI learning models, an AI prompt engineer should have a combination of technical, linguistic and analytical skills.

Recognized by the World Economic Forum as one of the top jobs of the future, a career in AI prompt engineering can be fruitful. The salary range for AI prompt engineers can vary significantly. Various sources mention salaries ranging from $175,000 to over $300,000. However, these figures are based on specific job listings and might not represent the entire range of salaries in the field. Additionally, salaries can vary based on factors such as geographical location, experience and the organization or industry hiring for the role.

All this may indicate an even more rapid development of the ability of artificial intelligence to understand user requests, in particular, to write program code for software development, and partly confirms the words of the CEO of Nvidia.

However, one should not assume that modern artificial intelligence technologies are capable of completely independently developing technically and logically complex software. Perhaps, through the efforts of researchers and developers, this day will come. However, at the moment, the use of artificial intelligence in software development is justified as an aid to novice developers and to facilitate routine processes for large IT companies.

References:

1. 2023 IT Job Market and BLS Data Analysis by Janco - Hiring of IT Pros increases (e-janco.com) *[Electronic resource]. – Mode of access:* https://e-janco.com/career/employmentdata.html. *– Date of access: 01.03.2024.*

2. 8 Ways To Use AI In Software Development For Max Efficiency | Zartis *[Electronic resource]. – Mode of access:* https://www.zartis.com/8-ways-to-use-ai-in-software-development*.html – Date of access: 01.03.2024.*

3. Jensen Huang says kids shouldn't learn to code — they should leave it up to AI | Tom's Hardware (tomshardware.com) *[Electronic resource]. – Mode of access:* https://www.tomshardware.com/tech-industry/artificial-intelligence/jensen-huang-advises-against-learning-to-code-leave-it-up-to*.html – Date of access: 01.03.2024.*