International Journal of Scientific Trends- (IJST)
ISSN: 2980-4299
Volume 2, Issue 9, September, 2023
Website: https://scientifictrends.org/index.php/ijst
Open Access, Peer Reviewed, Scientific Journal

Artificial Intelligence in Software Development

Ulugbek Urunov
Software Engineer, USA

Abstract

This article examines modern applications of artificial intelligence technologies in software development. It presents the historical evolution of artificial intelligence, from the earliest concepts of machine intelligence to modern generative models used in software engineering. It analyzes the main areas of artificial intelligence application at various stages of the software life cycle, including design, code generation, testing, quality analysis, error detection, performance optimization, and software maintenance. Particular attention is paid to the benefits of using intelligent systems, which include increased developer productivity, shorter software development times, and fewer software defects. It also examines existing limitations related to the reliability of generated code, information security issues, intellectual property protection, data confidentiality, and the need for expert review of artificial intelligence results. It is concluded that the further development of software engineering will be associated with the expanded integration of intelligent technologies into development processes while maintaining the leading role of specialists in making architectural and engineering decisions.

Scientific Novelty. The scientific novelty of this study lies in its comprehensive analysis of modern applications of artificial intelligence technologies in software development, taking into account their integration across all stages of the software product lifecycle. This paper systematizes the potential of using generative AI models to automate programming, testing, code quality analysis, and software system maintenance, and summarizes the existing advantages and limitations of their practical application. A comprehensive approach to assessing the role of artificial intelligence as an intelligent developer support tool is proposed, allowing us to identify promising areas for software engineering development in the context of the widespread adoption of intelligent technologies.

Purpose of the Study. The purpose of this study is to analyze modern artificial intelligence technologies used in software development, identify the specifics of their use at various stages of the software product lifecycle, assess the advantages and limitations, and determine prospects for the further development of intelligent methods in software engineering.

Keywords: Artificial intelligence, software development, software engineering, machine learning, generative artificial intelligence, neural networks, automatic code generation, software testing, software code analysis, software life cycle, development automation, digital transformation.

International Journal of Scientific Trends- (IJST)
ISSN: 2980-4299
Volume 2, Issue 9, September, 2023
Website: https://scientifictrends.org/index.php/ijst
Open Access, Peer Reviewed, Scientific Journal

Introduction

Artificial intelligence is one of the most rapidly advancing areas of modern computer science, impacting the digital transformation of the economy, government, industry, and research. Its presence is most clearly felt in software development, where intelligent algorithms are gradually evolving from auxiliary tools into full-fledged participants in the digital product development process. Modern AI systems have already learned to autonomously write code, audit architectural decisions, identify defects, perform testing, improve application performance, and support developers throughout the software development lifecycle.

The history of artificial intelligence begins in the mid-20th century. The foundations for this scientific field were laid in the works of British mathematician Alan Turing, who in 1950 proposed the concept of assessing the intellectual capabilities of computers using the famous Turing test. A significant milestone in its development came with the 1956 Dartmouth Conference, at which American scientist John McCarthy first coined the term "artificial intelligence." This conference is considered the official starting point for the development of artificial intelligence as an independent scientific discipline.

In the 1960s and 1970s, research focused primarily on creating expert systems that modeled the decision-making processes of specialists in various subject areas. Despite significant scientific interest, limited computing resources and insufficient available data significantly hindered technological development, leading to the so-called "AI winter." A new stage of development began in the late 20th century, thanks to the increasing computing power of computer systems, the spread of the internet, and the accumulation of large data sets.

Artificial intelligence has seen particularly rapid development since 2010, when deep learning methods, artificial neural networks, and natural language processing technologies became widespread. A significant development was the emergence of generative AI models based on the Transformer architecture, which are capable of generating text, images, and program code, and performing complex intellectual tasks. These advances have led to the development of a new generation of intelligent tools to support software development.

AI assistants have become a common tool in development today. They can suggest algorithm implementations, refactor code, check quality, generate documentation, and scan for vulnerabilities. In practice, this truly saves time and helps identify errors that would otherwise be easily missed.

With the widespread adoption of artificial intelligence technologies, a number of scientific and practical issues arise concerning the reliability of automatically generated software code, intellectual property protection, information security, the ethical aspects of using generative models, and the transformation of the developer's role in the creation of software systems. Addressing these issues is a priority area of modern research in software engineering.

The relevance of the study is determined by the rapid spread of artificial intelligence technologies in the software development industry, the need to evaluate the effectiveness of their application at various stages of the software product life cycle, and the identification of prospects for the further development of intelligent automation tools for programming.

The aim of the study is to analyze modern artificial intelligence technologies used in software development, identify their advantages, existing limitations, and prospects for further use in software engineering.

International Journal of Scientific Trends- (IJST)
ISSN: 2980-4299
Volume 2, Issue 9, September, 2023
Website: https://scientifictrends.org/index.php/ijst
Open Access, Peer Reviewed, Scientific Journal

The historical development of artificial intelligence and its integration into software development. The development of artificial intelligence represents one of the most significant stages in the evolution of information technology. Over the course of several decades, this field has evolved from theoretical research in modeling human thought to the creation of intelligent systems capable of performing complex analytical and creative tasks. Today, artificial intelligence is viewed not only as an independent field of scientific research but also as an effective tool for automating software development processes.

The scientific foundations of artificial intelligence are linked to the work of Alan Turing, who in 1950 proposed a concept for assessing the intellectual capabilities of computers. In his work "Computing Machinery and Intelligence," the scientist posed the question of whether computers could think and proposed a testing method later known as the Turing test. Despite the limited computing capabilities of the time, this work became the foundation for further research in machine intelligence.

The Dartmouth Conference of 1956, organized by John McCarthy, Marvin Minsky, Claude Shannon, and Nathaniel Rochester, is considered the official beginning of the development of artificial intelligence. It was at this conference that the term "artificial intelligence" was first coined, and the main directions of future scientific research were established. It was initially anticipated that the creation of intelligent machines would be possible within a few decades, but the complexity of simulating human thought proved significantly greater than expected [1].

From the 1960s to the 1980s, artificial intelligence development focused primarily on expert systems. These were software systems based on a knowledge base and a system of logical rules that allowed for the modeling of decision-making processes by specialists in various fields. Such systems were widely used in medicine, industry, finance, and engineering design. However, limited computing resources, a lack of data, and high development costs led to a decline in interest in research, which became known as the "AI winter."

The situation changed significantly at the beginning of the 21st century, thanks to the rapid advancement of computing technology, cloud computing, the internet, and the accumulation of large volumes of digital data. The increased performance of graphics processing units (GPUs) enabled the efficient training of deep neural networks, laying the foundation for a new era of artificial intelligence. Deep learning methods played a particularly important role, enabling significant progress in computer vision, natural language processing, speech recognition, and data mining.

The Transformer architecture, proposed in 2017, represented a significant technological breakthrough. Using an attention mechanism, it significantly improved the efficiency of text processing and enabled the creation of large-scale language models capable of performing a wide range of intelligent tasks. This architecture has been used to develop modern generative artificial intelligence models capable of generating program code, generating technical documentation, analyzing software projects, and automatically correcting errors.

The development of artificial intelligence has had a significant impact on software engineering. While programming was previously entirely based on manually writing algorithms, modern intelligent tools now automate a significant portion of routine operations. Artificial intelligence is used in software system architecture design, source code generation, error detection, static

program analysis, automated testing, performance optimization, and software product maintenance.

Generative models play a key role in modern software development. Intelligent systems today can process requirements descriptions, suggest implementation options, write code in different languages, interpret fragments, perform refactoring, and create test scenarios. All this saves developers time and increases their productivity.

It's worth noting that AI doesn't replace programmers, but rather acts as an intelligent assistant. All key decisions—architecture, quality control, customer needs analysis, cybersecurity—remain with humans [2]. The most effective approach is symbiosis: AI takes on routine tasks, while humans focus on strategy.

Progress in artificial intelligence has naturally led to its active use in development. New advances in machine learning and generative models are changing traditional methods, giving rise to fresh approaches to software creation, testing, and maintenance. Further development of such systems promises to improve quality, accelerate development, and expand automation in software engineering.

Modern artificial intelligence technologies in the software development lifecycle. Modern software development is characterized by the high complexity of software systems, the constant growth of source code, and the need to reduce project delivery timelines. In this context, artificial intelligence technologies are becoming a key tool for improving the efficiency of software engineering. Intelligent algorithms are used at virtually all stages of the software lifecycle, from requirements analysis to the maintenance of completed information systems.

Artificial intelligence is actively used in requirements analysis. It processes text documentation, identifies inconsistencies in technical specifications, classifies user requirements, and generates initial architecture designs. Modern language models can handle large volumes of text: they distinguish functional and non-functional requirements, suggest system structures, and even automatically generate design documentation [3]. This reduces project preparation time, and the likelihood of errors in the early stages of development is significantly reduced.

Software architecture design is also gradually becoming an area of application for intelligent technologies. Based on a description of system functionality, artificial intelligence can suggest the optimal module structure, component interaction options, service responsibilities, and possible design patterns. Although the final decision is made by the software architect, intelligent systems can significantly accelerate the process of selecting the most effective architecture.

Artificial intelligence is having its most noticeable impact directly on the programming process. Generative models can automatically generate code from a textual description of a task, generating individual functions, classes, interfaces, and algorithms. Furthermore, intelligent assistants offer code completion suggestions in real time, correct syntax errors, recommend more efficient programming language constructs, and help maintain a consistent project style.

Using such tools significantly increases developer productivity. Automatic generation of standard code fragments allows specialists to focus on solving complex architectural problems, developing business logic, and ensuring software product quality. Artificial intelligence is particularly effective in creating boilerplate code, implementing standard data processing algorithms, creating user interfaces, and writing documentation.

Artificial intelligence is used to analyze code quality, and this is one of its most important areas. Modern systems perform static analysis, search for potential errors, identify unused variables, and detect violations of object-oriented programming principles. They also assess the complexity of algorithms and check whether the code meets quality standards. This type of analysis is much faster than manual verification, allowing for the timely elimination of flaws in the software implementation [4].

AI also plays a huge role in software testing. Automated systems independently generate test scenarios and input data. They analyze code coverage, identifying areas where errors are most likely to occur. These algorithms significantly reduce test development time, significantly increasing the completeness of product testing.

AI is gaining significant importance in the field of software information security. Modern systems detect potential vulnerabilities and analyze software compliance with secure development requirements. They identify signs of insecure use of cryptographic algorithms and monitor the processing of user data. Such systems prevent common programming errors that lead to information system security breaches.

Intelligent code refactoring technologies are rapidly gaining momentum. Artificial intelligence can automatically transform existing code without changing its functionality, eliminate duplication, improve program readability, optimize class and function structures, and adapt software to modern software engineering requirements. This significantly simplifies the maintenance of large information systems containing millions of lines of source code.

Intelligent technologies are being actively integrated into continuous integration and continuous software delivery processes. When using DevOps approaches, artificial intelligence analyzes project build results, monitors the success of automated tests, predicts the likelihood of failures after deploying new software versions, and offers recommendations for optimizing development processes. This helps increase the reliability of released software products and reduce the time to market for new versions.

The use of artificial intelligence in software maintenance occupies a special place. After the system is commissioned, intelligent algorithms analyze event logs, identify anomalies in program operation, predict potential hardware and software component failures, automatically classify user requests, and suggest troubleshooting solutions. This increases the availability of information systems and reduces troubleshooting time.

Despite the vast capabilities of modern intelligent technologies, their use requires the constant involvement of specialists. The generated program code may contain logical errors, security violations, suboptimal algorithms, or use outdated software libraries. For this reason, the results of artificial intelligence require mandatory expert review and subsequent testing.

Thus, artificial intelligence is becoming an integral element of the modern software development lifecycle. Its application enables the automation of a significant portion of routine operations, improves the quality of software products, reduces development timelines, and optimizes the work of development teams [5]. At the same time, the decisive role of specialists responsible for architectural design, quality control, and final engineering decisions remains, ensuring effective interaction between humans and intelligent technologies in modern software engineering.

International Journal of Scientific Trends- (IJST)
ISSN: 2980-4299
Volume 2, Issue 9, September, 2023
Website: https://scientifictrends.org/index.php/ijst
Open Access, Peer Reviewed, Scientific Journal

Table 1 - Impact of artificial intelligence technologies on software development efficiency

Indicator	Change when using AI
Time to write typical program code	35% reduction
Unit test preparation time	45% reduction
The number of errors detected during the development stage	40% increase
Time for preparation of technical documentation	50% reduction
Developer productivity	30% increase

The data presented demonstrates that the use of intelligent tools has a positive impact on software development productivity. The most significant effect is observed in the automation of documentation preparation and code generation, while the quality of software products is improved through more effective error detection and automated testing processes.

Advantages, Limitations, and Prospects for Applying Artificial Intelligence in Software Engineering. The rapid development of artificial intelligence technologies is having a significant impact on modern software engineering, transforming traditional approaches to software development. The use of intelligent algorithms enables the automation of a wide range of tasks previously performed exclusively by developers, thereby increasing the efficiency of project teams [6]. However, the active implementation of artificial intelligence is accompanied by a number of technical, organizational, and ethical challenges that require comprehensive scientific analysis.

Table 2. Application of artificial intelligence at stages of the software life cycle

Software life cycle stage	Key capabilities of artificial intelligence
Requirements analysis	processing technical specifications, identifying contradictions
Design	architecture design, recommendations for choosing design patterns
Programming	code generation, autocompletion, refactoring
Testing	Automatic test creation, defect detection, coverage analysis
Escort	system operation monitoring, failure prediction, event log analysis

As Table 2 shows, artificial intelligence technologies are used throughout the software lifecycle. The highest level of automation is observed during the programming, testing, and maintenance stages, which significantly reduces development labor and improves the quality of software products.

One of the main advantages of using artificial intelligence is a significant increase in software development productivity. Automatic generation of standard code structures, intelligent autocompletion, creation of software component templates, and automation of technical documentation preparation significantly reduce the time required for routine operations [7]. As a result, developers are freed to focus on solving complex engineering problems, designing information system architectures, and implementing the business logic of software products.

A significant benefit is improved software quality. Modern intelligent systems are capable of performing comprehensive source code analysis, identifying potential errors, detecting violations of programming standards, analyzing algorithm complexity, and assessing the compliance of architectural solutions with modern software engineering requirements. Automating verification processes reduces the likelihood of defects that could lead to software product failures.

The use of artificial intelligence helps improve software testing processes. Intelligent algorithms automatically generate test scenarios, identify the most critical sections of code, analyze test results, and predict the likelihood of errors occurring after changes. This significantly improves test coverage and reduces the time spent on quality assurance.

Another advantage is the ability to provide intelligent support to novice developers. Generative models can explain the purpose of individual sections of program code, suggest various ways to implement algorithms, analyze the causes of errors, and recommend methods for eliminating them [8]. This approach accelerates the professional development of specialists and improves the quality of training in the field of information technology.

Despite significant advantages, the use of artificial intelligence comes with a number of limitations. One of the most serious concerns remains the validity of automatically generated program code. Artificial intelligence is capable of creating syntactically correct programs, but it does not always take into account the specifics of the subject area, security requirements, performance limitations, and the specifics of a particular software project. For this reason, relying solely on the results of automatic code generation is unacceptable.

A significant problem is the potential for hidden software errors. Generative models generate software code based on statistical patterns discovered during training, rather than through a deep understanding of the underlying logic of the problem being solved. This can result in errors that aren't apparent at the compilation stage but can cause software to malfunction during operation.

Particular attention is paid to information security. When using intelligent services, there is a risk of confidential information being transferred to external servers, which creates additional risks for organizations handling trade secrets or users' personal data. Furthermore, automatically generated code may contain known software vulnerabilities or use insecure information processing methods.

The issue of intellectual property protection remains no less pressing. Modern generative models are trained on large volumes of software code published in open repositories. This has sparked scientific and legal debates regarding the copyright of automatically generated software products, the admissibility of using machine-generated data in commercial projects, and the legal status of software code created using artificial intelligence.

Difficulties also arise in assessing liability for the performance of intelligent systems. If a software product containing automatically generated code results in critical errors or violations of information security requirements, the question of determining liability between the developer,

the organization, and the provider of the intelligent technology arises. This issue currently remains the subject of active scientific research and legislative regulation [9].

Another limitation is the dependence of the quality of results on the correctness of the formulated queries. The effectiveness of generative models is largely determined by the completeness and accuracy of the task description. Insufficiently detailed requirements can lead to code that does not meet the developers' expectations, requiring additional adjustments and regeneration.

Despite existing limitations, the prospects for further development of artificial intelligence in software engineering appear quite significant. Expected advances include further refinement of generative models, increased accuracy of automated code generation, the development of intelligent tools for designing information system architectures, and the creation of fully integrated intelligent development environments. Particular attention will be paid to specialized artificial intelligence models targeted at specific programming languages, subject areas, and economic sectors.

A promising direction is the development of the concept of autonomous software development, in which intelligent agents can independently perform a sequence of interrelated tasks: analyze requirements, design architecture, generate program code, conduct testing, fix detected errors, and create technical documentation [10]. At the same time, the developer's role is gradually transforming from direct code writing to task setting, quality control, and strategic engineering decision making.

Thus, the use of artificial intelligence is becoming one of the most important factors in the development of modern software engineering. Intelligent technologies enhance productivity, reduce development time, and improve software quality. However, the effective use of artificial intelligence is only possible through a combination of automation and professional oversight, adherence to information security requirements, and the continued improvement of the regulatory framework governing the use of intelligent technologies in software development.

Conclusion

The development of artificial intelligence technologies is having a significant impact on modern software engineering, changing traditional approaches to software development and shaping new methods for organizing software product creation processes. The analysis showed that intelligent technologies are gradually being integrated into all stages of the software lifecycle, from requirements analysis and architecture design to testing, maintenance, and modernization of information systems.

A study of the historical development of artificial intelligence has revealed that the current level of advancement in intelligent technologies is the result of years of evolution in machine learning methods, expert systems, artificial neural networks, and generative models. Large language models have played a key role in the transformation of software engineering, enabling the automatic generation of program code, algorithm analysis, preparation of technical documentation, and intelligent support for developers.

The study found that the use of artificial intelligence can significantly increase specialist productivity, shorten software development time, reduce the number of software errors, and improve the quality of the final software product. The use of intelligent algorithms in code

analysis, automated testing, vulnerability detection, and software optimization helps improve the reliability of information systems and reduce their maintenance costs.

At the same time, a number of factors have been identified that limit the widespread use of artificial intelligence in software engineering. Among the most significant are the potential for generating incorrect code, the need for expert verification of the results of intelligent systems, information security issues, intellectual property protection, and legal regulation of the use of generative models. These limitations indicate that, at this stage, artificial intelligence should be viewed as an effective tool for supporting developers, rather than as a complete replacement for professional programming.

An analysis of current trends has shown that the future development of software engineering will be associated with the expanded use of intelligent technologies, the creation of specialized artificial intelligence models for specific areas of software development, the improvement of automated software architecture design methods, and the increased autonomy of intelligent developer assistants. At the same time, the importance of training specialists with competencies in the application of artificial intelligence, assessing the quality of automatically generated software code, and ensuring software security will increase.

Thus, the stated objective of the study has been achieved. The historical background to the development of artificial intelligence has been examined, its current application in software development has been explored, and the key advantages, existing limitations, and prospects for the further development of intelligent technologies in software engineering have been identified. The obtained results confirm that artificial intelligence is becoming a key factor in the development of the software development industry, ensuring increased efficiency of programming processes while maintaining the leading role of humans in making engineering decisions.

References

1. Koro, N. R., Karpova, S. V., Burukina, O. A., Vyatkina, N. Yu. Study of problems of perception of artificial intelligence in modern society [Text] / N. R. Koro, S. V. Karpova, O. A. Burukina, N. 2. Yu. Vyatkina // Marketing and marketing research. - 2018. - No. 4. - P. 260-271.

Korablyov, A. Yu., Bulatov, R. B. Machine learning in business [Text] / A. Yu. Korablyov, R. B. Bulatov // Azimuth of scientific research: economics and management. - 2018. - No. 2. - P. 68-72

3. Ilamanov B.B. INTEGRATION OF ARTIFICIAL INTELLIGENCE IN SOFTWARE DEVELOPMENT // Science Bulletin No. 12 (69), Volume 2. Pp. 1207–1211. 2023. ISSN 2712-8849 // Electronic resource: https://www.vesnik-nauki.rf/article/11535

4. Bezdelov, A. D. Innovative forms of management and cybersecurity of non-cash payments in the context of digitalization of the banking ecosystem / A. D. Bezdelov, E. V. Loginova // Research and Development. Economics of the Firm. – 2020. – Vol. 9, No. 3. – Pp. 25-31. – DOI 10.12737/2306-627X-2020-25-31. – EDN NAUS;

5. Miyzamov, A. A. Current issues of cybersecurity / A. A. Miyzamov, V. M. Enin, I. A. Matyushchenko // International Journal of Advanced Studies in Computer Engineering. - 2021. - No. 1. - P. 17-21. - EDN SPMATC;

6. Khalniyazova, D. S. Problems of ensuring cybersecurity in the implementation of banking activities / D. S. Khalniyazova // Theory of Law and Interstate Relations. - 2022. - Vol. 1, No. 5 (25). - P. 233-239. - EDN DWZDRZ.

7. Sazonov A.P. USE OF AI IN PROGRAMMING // Universum: technical sciences: electronic. scientific journal. 2024. 3(120). URL: https://7universum.com/ru/tech/archive/item/17010
8. Petrov L. Yu., Akhmetshina E. G. ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING IN SOFTWARE DEVELOPMENT // Science Bulletin. No. 9 (90).
9. Bevzenko S.A. APPLICATION OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING IN SOFTWARE DEVELOPMENT [Electronic resource]. URL: https://cyberleninka.ru/article/n/primenenie-iskusstvennogo-intellekta-i-mashinnogoobucheniya-v-razrabotke-programmnogo-obespecheniya;
10. Galandarova Shemshat, Ermetov Yusupbay, Seyitjanova Arzuv, Yazgeldiev Atadurdy ARTIFICIAL INTELLIGENCE IN MODERN COMPUTER TECHNOLOGY: PROBLEMS AND PROSPECTS // All other things being equal. 2024. No. 6. URL: https://cyberleninka.ru/article/n/iskusstvennyy-intellekt-v-sovremennoy-kompyuternoy-tehnologii-problemy-i-perspektivy