Secure Systems Design and Technology Development. Veliko Tarnovo, Bulgaria
Proceedings of the International Scientific Conference, 2026. Volume I, 203-208

AI-Driven Model for Economic Efficiency in Software Development Tasks

Viktor Hristov
Management Department
University of National and World Economy
Sofia, Bulgaria
viktor.hristov@unwe.bg

Miglena Angelova
Management Department
University of National and World Economy
Sofia, Bulgaria
m.angelova@unwe.bg

***Abstract*— The use of Artificial Intelligence (AI) has become an inseparable part of the software development process, parts of which are routine, requiring precision rather than creativity. AI code generation speeds up productivity by automatically completing code fragments and frees up time and brainpower to perform more tasks that require creativity or problem-solving. When writing code, the programmer uses not only the accumulated experience, but often also intuition, which helps him assess both the correctness of the code and the implications of its use.**

At the same time, AI uses different and increasingly sophisticated models and a thought chain resembling human thinking through modern approaches to Large Language Models (LLM) and Neural Networks. However, increasingly serious problems are emerging when using AI.

The publication proposes a model of an AI algorithm with two functionally different cores: one creative and one process, including control functions. This model overcomes a large part of the identified shortcomings of AI and complements their development. The model is supplemented with quantitative metrics to determine the economic efficiency of its application.

Keywords*— *Artificial Intelligence, Economic Effectiveness, Two core AI model

I. INTRODUCTION

AI becomes increasingly integral part of activities in software development companies. It spreads over vast activities in Software Development Life Cycle and automatizes activities like Project Management, Coding and Quality Assurance. Still efficiency of AI assisted or committed tasks is yet to be confirmed.

II. LITERATURE REVIEW

The broad academic discussion about the benefits of using Artificial Intelligence is extremely lively and has enjoyed the attention of researchers in recent years, especially after the widespread penetration of AI into work and personal life. Respectively, the use of AI in software development and work on software tasks is also an issue on which a multi-layered discourse is currently underway. Scientists are studying various aspects - in terms of efficiency and productivity, achieved level of quality and reliability, security and potential risk, interaction between humans and AI, management and decision-making, Algorithmic Management, purely economic dimensions of AI in software development, etc.

In this context, the role of artificial intelligence extends beyond isolated applications and increasingly encompasses the full lifecycle of software development. Revuri, Sakthivel and Nagasubramanian [1] acknowledge that artificial intelligence is radically transforming software development, introducing a higher degree of automation, which in turn leads to increased productivity. The scholars argue that through applications such as automatic code generation, error detection and test creation, AI improves the efficiency and quality of development at all stages of the SDLC, while preserving the primary role of human creativity.

This systemic integration of AI naturally leads to transformations not only in processes but also in the way software development activities are structured and executed. De La Cruz et al. [2] state that artificial intelligence is radically changing the way programmers work. The increasing and increasingly ubiquitous use of AI tools naturally requires the possession and development of new skills, instead of just focusing on writing code, as before. At the same time, scientists explicitly specify that these tools are not completely reliable and can lead to errors, so it remains important for a person to check and control the results.

According to a study by Kumari et al. [3], artificial intelligence is integrated into all stages of software development, with the main advantages being that it automates routine activities and contributes to increasing productivity, quality, and speed of development. They see

Online ISSN 3044-7771
https://doi.org/10.68302/std2026.vol1.177

AI as a major opportunity to increase the efficiency of modern software processes.

Beyond general improvements in productivity and speed, more granular analyses focus on the specific mechanisms through which these effects are achieved. Adnyana and Schwung [4] comment that the application of generative AI models to the creation of program code leads to a significant reduction in the effort required for subsequent corrections and to an improvement in the quality of the results. In addition, it is highlighted that the way the instructions are formulated has a greater impact on the accuracy and reliability of the generated code than the choice of a specific model.

Complementing these findings, other studies emphasize that the effectiveness of AI is not uniform, but contingent upon technological and organizational alignment. Banh, Holldack, and Strobel [5] argue that generative AI has the potential to increase productivity in software development. However, they emphasize that its successful implementation depends on the alignment of technological capabilities, organizational practices, and the way developers themselves work.

This line of reasoning is further reinforced by comparative analyses of different types of AI solutions. According to Bañuls-Lapuerta et al. [6], the effectiveness of generative AI models in software creation varies significantly depending on the type of model used, with paid solutions demonstrating higher accuracy and less need for additional interactions.

In addition to general development processes, AI demonstrates strong applicability in specific functional domains of software engineering. According to Czarnacka-Chrobot [7], the application of artificial intelligence in the processes of measuring the functional size of software leads to a reduction in subjectivity and labour intensity, while improving the accuracy and reproducibility of estimates. This creates the real prerequisites for more reliable planning of the efforts, costs and duration of software projects.

Similarly, the role of AI becomes particularly visible in higher-level decision-making processes within software development. Esposito et al. [8] claim that generative AI has a very good application in software architecture processes, especially in making architectural decisions and transforming requirements into code. However, they also note that there is currently a lack of established methods for evaluating the results, which limits the possibilities for objectively assessing its effectiveness.

However, alongside these benefits, the integration of AI introduces a set of critical challenges related to security and reliability. According to Glas et al. [9], generative artificial intelligence increases the efficiency of software development, but at the same time creates significant security risks, mainly related to the unreliability of the generated code and excessive trust of developers. Scientists are of the opinion that its effective use requires a human-centred approach that allows for the use of the technological capabilities of AI but retains human control and critical assessment.

These concerns are further expanded in studies focusing on secure software development practices. Al-Hashimi et al. [10] conclude that generative AI is increasingly being used in practice as a means of improving security in the development process, especially by detecting vulnerabilities and supporting secure programming. On the other hand, researchers also pay significant attention to the fact that its implementation creates new challenges related to reliability, ethical risks, and integration into existing processes.

At a more advanced level, the discussion extends to the need for explainability and transparency in AI-driven systems. Basheer et al. [11] comment that modern approaches to vulnerability detection using artificial intelligence place emphasis not only on accuracy. The same importance should be paid also on explainability, reliability, and security of models. This leads to the conclusion that effective application of AI in software development requires a professional combination of high performance with transparency and control.

Despite the increasing level of automation, the human factor remains central in determining the effectiveness of AI applications. In this line of consideration, it should be noted that a study by Shao and Ishengoma [12] shows that the use of generative artificial intelligence in software development depends largely on people – on their experience, attitudes and the influence of the environment around them. Furthermore, there are clear differences between the way students and professionals work with these tools, which in turn leads to the conclusion that the effects of AI are not the same for everyone.

This observation aligns with comparative perspectives on human–AI collaboration. Ampatzoglou et al. [13] conduct a comparative analysis between AI-assisted and fully human solutions. They come to the idea that there is a clear distinction between the two approaches: AI significantly supports the process for more precise task execution, but on the other hand, human intervention remains indispensable in recognizing problems and making quality decisions.

At the same time, the current stage of development of AI-based systems reveals a number of structural limitations. Giray [14] concludes that the development of machine learning-based systems is characterized by a high degree of complexity and non-determinism, with a lack of established engineering practices and tools in several areas. It is also found that despite growing interest, existing solutions are often at an early stage of development and have not been sufficiently validated in a real environment.

From a broader organizational perspective, the application of AI extends beyond software development to encompass core business processes. Anguelov [15] takes a more conceptual approach to the use of AI in business organizations. According to the author, the application of artificial intelligence in enterprise resource management systems covers a wide range of organizational processes, including supply chain management, financial activities, and human resources. Therefore, it can be concluded that

AI is used as a tool for optimization and automation of various business functions.

Within this evolving landscape, the question of how to evaluate the efficiency of AI-supported software development tasks becomes increasingly significant, particularly in the context of model-based and predictive approaches. In the development of AI-based models, the essential role of the indicators used is highlighted. According to Anguelov [16], their selection directly affects the accuracy of the results. In this logical sequence of reasoning, it can be assumed that in the assessment of the effectiveness of software tasks, the selection and structuring of indicators is decisive for the reliability of the predictive function of the model.

The aforementioned studies clearly show that artificial intelligence can significantly increase the efficiency of software development. However, it should also be noted that its impact is mediated by technological, organizational and human factors. This necessitates the development of structured, indicator-based approaches to ensure the reliability of the predictive function of AI-powered models.

III. AI SPECIFICS AND WEAKNESSES

Machine Learning (ML) process has following group of activities: *Model Requirements*, *Input data* (collection/preparation/processing), *Feature engineering*, *Model training and evaluation*, *Deployment and maintenance* [17]. Input data and the phenomenon of data cascades show theirs' elevating importance to AI efficiency [18]. It can also lead to inaccurate and biased results [19] or to spread in the model errors like so called silent failures [20]. Improving data quality should be done if possible before the data is inserted into AI model. Furthermore, internal mechanism for data correction should be available, which for AI based on Neural Network (NN) system is a process called backpropagation.

There are many scientific articles which point to the central role of learning algorithms in AI systems. Few authors [21], [22], [23] point out in different ways importance of learning algorithm to AI. Core of learning methodology and algorithms lies in different theories, most of them focused on the ability of the model to predict accurate as possible results and optimization of that process. Empirical risk minimization [24] or Bias-Variance Tradeoff [25] are even more directly aimed on that. There are additional optimization theories like Convex Optimization Optimality [26], [27], which address optimal (shortest) paths for dynamic programming. Reinforcement Learning and Bayes Optimality (Probabilistic Learning) offers decision rules under uncertain environment. Emphasis on all of them is to minimize errors and predict accurate as possible results. However, that process could slow down performance and does not consider efficiency. Successful implementation of learning algorithms depends on the supervision of learning process, which can be done via human interactions. This process is known as Human in the loop (HITL). HITL implementation in Reinforcement learning model brings valuable contribution.

All these methods are essential to learning algorithm optimization and reducing the errors, but it does not consider how effective outputs are. We acknowledge that some tasks in software development are still handled more effectively by humans, for example: naming conventions and redefinition of variables, more high severity errors in the code uncomplete code documentation [28]. Y. Liu et al. [29] are pointing that it is "very unlikely" all AI generated code has been reviewed by human, and thus there could be unrevealed errors, including security vulnerabilities. Above authors' analysis is discovering three types of violation done by AI assisted coding: *Code smells*, *Runtime bugs* and *Security issues*, with comparison results that AI assisted coding can reduce the time needed to fix maintenance issues and to generate simple and repetitive patterns. Committed field experiment also confirmed that for "easy and medium-level problems" AI enhances the performance of coding solutions compared to novice programmers' output [30].

As one of the conclusions of challenges in front of AI assisted systems marked above, we see that sometimes it is more efficient to use human resources or to put humans in the loop instead of relaying on AI fully automated processes. If that interaction is unnecessary often it will decrease overall efficiency of AI implementation. In addition, drawbacks that put obstacles across successful AI implementation are:

- *Underestimated optimum properties in terms of economic efficiency*
- *Possibility for peak loading and performance degrade*

IV. AI TWO CORE CONCEPT

A. Description

The AI two core concept encompasses first core conditionally named *Creative*, and second core named *Operative*. First core is dedicated to creative intensive tasks it is based on modern learning techniques. Second core has variable sets of rules and dictionaries, and it will maintain control functions as well. From another point of view Creative core can be seen as microprocessor maximum configuration, Operative – as minimal [31]. Based on stochastic and classical learning model Operative core maintains stable sets of proven solutions, optionally some of them marked with calculated economic efficiency coefficient (*EEC*) [32]. Creative core is designed to pass all prompts to Operative Core. If results are already available there Operative Core will return them back together with optionally calculated *EEC*, and it will do validation operations to avoid violations (code review for example).

Every user prompt goes to Creative core that automatically passes it to Operative Core which holds previous results and *EEC* [33]. It also monitors learning and processing operations in Creative Core, and if its resources are fully loaded or prompts are repeating, it triggers automatically human interventions. Operative core has the duty to avoid repeating answers or hallucinations

possibly done by Creative Core and automatically supervise its answers, and vice versa – Creative Core validates all results generated by Operative core. If there is a conflict, orchestrator triggers HITL interventions. Operative core also stores the data about previously executed tasks marked as effective economically with their respective *EEC* [34].

General diagram of the concept is presented on "Fig. 1" and descriptions of some characteristics are noted in "Table 1".

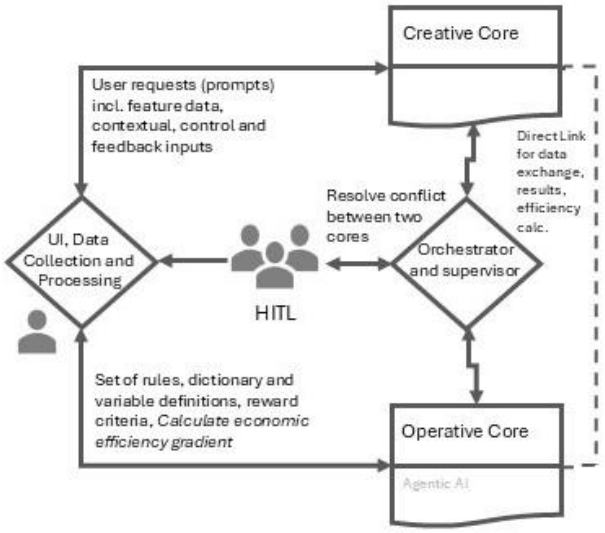

Fig. 1. Two core concept diagram

Sometimes learning and prediction could tend to use enormous learning steps, with very large number of passes. Learning algorithms could also tend to be unvaluable from economic perspective. To avoid that, authors propose to use dynamically calculated dedicated bias coefficient incorporated in all learning algorithms - *EEC*. The role of *EEC* is to put conditions to pass or prevent upon AI decisions. It is designed to adjust boundaries of the decision just like regular bias coefficient, with the difference it will be dedicated only for the purpose and set as optional attribute. In other words, it is bias factor which will decide which task to be executed by AI and which by human. It can also serve to distribute work between two cores and to assign more resources to economically valuable user prompts.

B. Use of EEC

EEC is calculated for functions with low level of Conditional entropy ($H(Y \mid x) \rightarrow min$). Smallest independent unit of work done by AI should have its own economic coefficient c that is obtained from previous experience (or learning cycle). By independent unit we understand that all its elements or components have equal significance by each other. *EEC* is a sum of all economic coefficients (c_n) managed for more complex tasks or project phases. It can be designed to be calculated by ML with HITL interactions. The values of c can be adjusted based on the nature of the task and/or previous knowledge and experience.

$$eec = c_1 + c_2 + \cdots c_n \quad (1),$$

where c_n is indexed economic coefficient of smallest independent task

EEC is integrated in learning algorithms, for example of backpropagation:

$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + {}_meec \quad (2)$$

$$if\ z > R\ then\ a = f(z)\ else\ null\ end \quad (3),$$

where z – computational unit; x – inputs, w – weights, a, $f(z)$ – activation function, R – reference value, ${}_meec$ – normalized value of *eec*.

Normalization rules need to be defined later, but as general guideline authors propose weight of *eec* should be no more than 15% of value of backpropagation equation. Populated value of z is used only to compare with human defined boundaries and trigger or reject activation function. The more complex the task is, the more efficient will be the algorithm.

Example: we have AI assisted development task A consisted of two subtasks A1 (generate drop-down menu in UI) with $c = 5$ and A2 (button with submit action) with $c = 7$. Coefficients are calculated based on previous experience and c is ranked [1..10], 1 - means very ineffective and needs a consideration to be done solely by human, 10 - means perfectly executed by AI. The assessment is done entirely from economic point of view. *EEC* for task A is calculated as average value of A1 and A2 (equal significance). *EEC = 6.5*. This value is added into backpropagation after normalization. We know from previous experience that z minus *eec* is 60. We set R for A task at level 70. That means activation function will not be triggered, and we change x inputs and their respective w in order to achieve economic efficiency and trigger activation function. During the time x and w can be changed, *eec* will remain the same unless specific input for it is provided by humans or dedicated learning algorithm. In other case z can become "economic ineffective" and inputs needs to be changed anyway (for example do not ask AI to create button with specific action).

Activation function is enabled only if neuron value is above parametrized boundary values stored in Operative core.

TABLE 1 CONCEPT MAIN ELEMENTS OVERVIEW

Two Core Model	***Purpose***	***Learning Methods***	***Expected output***	***Data source***	***Human Supervised***
Creative Core	Handle all creative tasks, continuous learning with human supervised Deep Learning and Neural Network models	Supervised fine-tuning; Reinforced Learning from Human Feedback; Preference Optimization; Diversity-aware preference; Generative Adversarial Networks	Creative, out of the box solutions; complex tasks solution; code new features	Live data, Human labelled data, adaptive data modification	Yes, during learning process
Operational Core	Handle all repeating complex or all simple tasks, integrated	Linear Regression or Classification; Support Vector Machine (SVM);	Check and validate code, monitor and distribute	Verified results from Creative Core;	No, only parameters modification

Two Core Model	***Purpose***	***Learning Methods***	***Expected output***	***Data source***	***Human Supervised***
	supervisory controls, KPIs, AI agents for task distribution, calculating R	Kernel SVM; k-Nearest neighbor; Decision Tree	tasks between teams, Compute control samples	stores all data about definitions and variables	

C. Advantages of the concept

- *Enable secure validation of AI output*: one core can be more open to getting large and complex data with state of art algorithms thus more exposed to external threats, while second one can be more encapsulated which will directly control output of the first core and in case of colliding or conflicting results to trigger human in the loop interventions. As an example, AI tend more to hard code API passwords which can be fixed directly during code generation by operational core.

- *Ensure monitoring of economic effectiveness*: real-time direct supervision of results and controlling.

- *Distribute work in efficient and programable way*: tasks are passing through creative core, which is handling only complex and generative ideas, the rest like daily work distribution are passed to Operative core.

- *Reduce human supervision and promote it as final control point*: it provides substitution for some of human supervised learning tasks.

V. CONCLUSION

AI systems create opportunities to increase the performance and reduce the cost of all type of tasks, but the successful implementation requires strategic and focused approach. Close monitoring, embedded controls and specific value-driven metrics are crucial. Proposed concept tries to address those issues, though it requires per case adaptation and empirical validation. It provides different view of distributed resources aimed to promote AI self-supervision along with economic metrics incorporated into AI learning logic.

ACKNOWLEDGEMENTS

The research is funded by the National Science Fund of Bulgaria under the project KP-06-N55/3 on the topic “Conceptual Model for Evaluation of Fast-Growing Companies Operating in Innovation-Intensive Industries Based on Artificial Intelligence Methods”.

REFERENCES

[1] J. Revuri, R. K. Sakthivel, G. Nagasubramanian, Chapter Five - Artificial intelligence (AI) technologies and tools for accelerated software development, Editor(s): Pethuru Raj, Mats Agerstam, Pushan Kumar Dutta, B. Sundaravadivazhagan, Gayathri Nagasubramanian, Advances in Computers, Elsevier, Volume 141, 2026, Pages 115-159, ISSN 0065-2458, ISBN 9780443224010, https://doi.org/10.1016/bs.adcom.2025.07.001.

[2] E. De La Cruz, H. Le, K. Meduri, G. S. Nadella, H. Gonaygunta, Redefining the Programmer: Human-AI Collaboration, LLMs, and Security in Modern Software Engineering, Computers, Materials and Continua, Volume 85, Issue 2, 2025, Pages 3569-3582, ISSN 1546-2218, https://doi.org/10.32604/cmc.2025.068137.

[3] K. S. Kumari, B. Sundaravadivazhagan, V. Indumathy, D. Maladhy, Chapter Seven - Detailing AI techniques and tools for software engineering acceleration and automation, Editor(s): Pethuru Raj, Mats Agerstam, Pushan Kumar Dutta, B. Sundaravadivazhagan, Gayathri Nagasubramanian, Advances in Computers, Elsevier, Volume 141, 2026, Pages 183-209, ISSN 0065-2458, ISBN 9780443224010, https://doi.org/10.1016/bs.adcom.2025.07.007.

[4] K. Adnyana, A. Schwung, Benchmarking and validation of prompting techniques for AI-assisted industrial PLC programming, Machine Learning with Applications, Volume 23, 2026, 100804, ISSN 2666-8270, https://doi.org/10.1016/j.mlwa.2025.100804.

[5] L. Banh, F. Holldack, G. Strobel, Copiloting the future: How generative AI transforms Software Engineering, Information and Software Technology, Volume 183, 2025, 107751, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2025.107751.

[6] F. A. Bañuls-Lapuerta, V. Marti-Miralles, R. J. Gónzalez-García, G. Martínez-Rico, Using Natural Language Prompts With AI Models for Low-Cost Assistive Software Design: Exploratory Comparative Evaluation, JMIR Rehabilitation and Assistive Technologies, Volume 13, 2026, ISSN 2369-2529, https://doi.org/10.2196/86786.

[7] B. Czarnacka-Chrobot, AI applications in software functional size measurement: A systematic literature review, Journal of Systems and Software, Volume 232, 2026, 112686, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2025.112686.

[8] M. Esposito, X. Li, S. Moreschini, N. Ahmad, T. Cerny, K. Vaidhyanathan, V. Lenarduzzi, D. Taibi, Generative AI for software architecture. Applications, challenges, and future directions, Journal of Systems and Software, Volume 231, 2026, 112607, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2025.112607.

[9] M. Glas, C. Nirschl, B. Lanyado, J. van Niekerk, Insecure by design? A human-centric security perspective on AI-assisted software development, Computers & Security, Volume 164, 2026, 104842, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2026.104842.

[10] H. A. Al-Hashimi, R. A. Khan, H. S. Alwageed, A. M. Algarni, S. Ayouni, A. O. Almagrabi, Exploring the role of generative AI in enhancing cybersecurity in software development life cycle, Array, Volume 28, 2025, 100509, ISSN 2590-0056, https://doi.org/10.1016/j.array.2025.100509.

[11] N. Basheer, S. Islam, P. Kumar, D. Javeed, N. Islam, A responsible AI-driven framework for robust and transparent software vulnerability detection, Information and Software Technology, Volume 195, 2026, 108126, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2026.108126.

[12] D. Shao, F. Ishengoma, Empirical analysis of generative AI tool adoption in software development, Information and Software Technology, Volume 192, 2026, 108036, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2026.108036.

[13] A. Ampatzoglou, E.-M. Arvanitou, S. Almpanopoulos, N. Mittas, A. Chatzigeorgiou, AI-assisted code refactoring: Where can it be helpful and where do humans outperform it?, Journal of Systems and Software, Volume 238, 2026, 112862, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2026.112862.

[14] G. Giray, A software engineering perspective on engineering machine learning systems: State of the art and challenges, Journal of Systems and Software, Volume 180, 2021, 111031, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2021.111031.

[15] K. Anguelov, "Applications of Artificial Intelligence for Optimization of Business Processes in Enterprise Resource Planning Systems," 2021 12th National Conference with International Participation (ELECTRONICA), Sofia, Bulgaria, 2021, pp. 1-4, https://doi.org/10.1109/ELECTRONICA52725.2021.9513677

[16] K. Anguelov, "AI-Based Predictive Model for High-Growth Enterprises," 2025 13th International Scientific Conference on Computer Science (COMSCI), Sozopol, Bulgaria, 2025, pp. 1-5, https://doi.org/10.1109/COMSCI67172.2025.11225285.

[17] Z. Fang, Y. Yuan, J. Zhang, Y. Liu, Y. Mu, Q. Lu, X. Xu, J. Wang, Ch. Wang, Sh. Zhang, Sh. Chen, MLOps Spanning Whole Machine Learning Life Cycle: A Survey, arXiv, arXiv:2304.07296, 2023, https://doi.org/10.48550/arXiv.2304.07296.

[18] N. Sambasivan, Sh. Kapania, H. Highfill, D. Akrong, Pr. Paritosh, L.M. Aroyo, "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI, CHI '21: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, Article No.: 39, Pages 1 – 15, 2021, NY, USA, ISBN 9781450380966, https://doi.org/10.1145/3411764.3445518.

[19] D. Zha, Z. P. Bhat, K. Lai, F. Yang, Zh. Jiang, Sh. Zhong, X. Hu, Data-centric Artificial Intelligence: A Survey, arXiv, arXiv:2303.10158, 2023, https://doi.org/10.48550/arXiv.2303.10158.

[20] N. Polyzotis, S. Roy, St. E. Whang, M. Zinkevich, Data Lifecycle Challenges in Production Machine Learning: A Survey, Association for Computing Machinery, SIGMOD Rec. Volume 47, 2018, ISSN 0163-5808, https://doi.org/10.1145/3299887.3299891.

[21] H. Naveed, Ch. Arora, H. Khalajzadeh, J. Grundy, O. Haggag, Model driven engineering for machine learning components: A systematic literature review, Information and Software Technology, Volume 169, 2024, 107423, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2024.107423.

[22] Ch. Janiesch, P. Zschech, K. Heinrich, (2021). Machine learning and deep learning. 10.48550/arXiv.2104.05314, https://doi.org/10.1007/s12525-021-00475-2.

[23] Y. Watanabe, H. Washizaki, K. Sakamoto, D. Saito, K. Honda, N. Tsuda, Y. Fukazawa, N. Yoshioka, (2019). Preliminary Systematic Literature Review of Machine Learning System Development Process. 10.48550/arXiv.1910.05528, https://doi.org/10.48550/arXiv.1910.05528.

[24] O. Bousquet, A. Elisseeff, (2002). Stability and Generalization. Journal of Machine Learning Research. 2. 499-526. https://doi.org/10.1162/153244302760200704.

[25] B. Neal, S. Mittal, A. Baratin, V. Tantia, M. Scicluna, S. Lacoste-Julien, I. Mitliagkas, (2018). A Modern Take on the Bias-Variance Tradeoff in Neural Networks. 10.48550/arXiv.1810.08591, https://doi.org/10.48550/arXiv.1810.08591.

[26] R. M. Freund, Introduction to Optimization, and Optimality Conditions for Unconstrained Problems, Massachusetts Institute of Technology, OpenCourseWare, 2004, https://ocw.mit.edu/courses/15-084j-nonlinear-programming-spring-2004/f8e5bd09d99cc296e0f5af25bf607e49_lec1_unconstr_opt.pdf

[27] R.L. Carraway, T.L. Morin, Theory and applications of generalized dynamic programming: An overview, Computers & Mathematics with Applications, Volume 16, Issues 10–11, 1988, Pages 779-788, ISSN 0898-1221, https://doi.org/10.1016/0898-1221(88)90188-5.

[28] Sh. A. Licorish, A. Bajpai, Ch. Arora, F. Wang, K. Tantithamthavorn, Comparing Human and LLM Generated Code: The Jury is Still Out!, arXiv, arXiv:2501.16857, 2025, https://doi.org/10.48550/arXiv.2501.16857

[29] Y. Liu, R. Widyasari, Y. Zhao, I. Cl. Irsan, D. Lo, Debt Behind the AI Boom: A Large-Scale Empirical Study of AI-Generated Code in the Wild, arXiv, arXiv.2603.28592, 2026, https://doi.org/10.48550/arXiv.2603.28592.

[30] N. Nascimento, P. Alencar, D. Cowan, Comparing Software Developers with ChatGPT: An Empirical Investigation, arXiv, arXiv:2305.11837, 2023, https://doi.org/10.48550/arXiv.2305.11837

[31] R. Petrov, Mikroprotsesorni sistemi i mikrokontroleri: Uchebno pomagalo za zadalzitelna podgotovka v profesionalnite gimnazii s napravlenie „Elektronika i avtomatizatsiya". Sofia: Novi znaniya, 2011, ISBN 978-954-2907-04-6

[32] Luchkov, K., & Velinova-Sokolova, N. (2026). Methodology for Quantitative Risk Assessment in the Integration and Use of ERP Systems in Enterprises. Journal of Risk and Financial Management, 19(3), 226. https://doi.org/10.3390/jrfm19030226

[33] Luchkov, K., Chipriyanova, G., Chipriyanov, M. (2025). Implementing Innovative Strategies for Competitiveness in the Era of Digital Transformation. In: Alzoubi, H.M., Cheng Ling, T., El Khatib, M. (eds) Innovative Dynamics in Management and Engineering (IDME) . ISCME 2024. Advances in Science, Technology & Innovation. Springer, Cham. https://doi.org/10.1007/978-3-031-90131-7_29

[34] Chipriyanova, G., Chipriyanov, M., Luchkov, K. (2024). Investigation and Analysis of Attitudes Towards the Implementation of Artificial Intelligence in Internal Business Processes. In: Proceedings of the 15th International Scientific and Practical Conference "Environment, Technology and Resources", Rezekne, 2024, Vol. II, pp. 61–67. https://doi.org/10.17770/etr2024vol2.8032