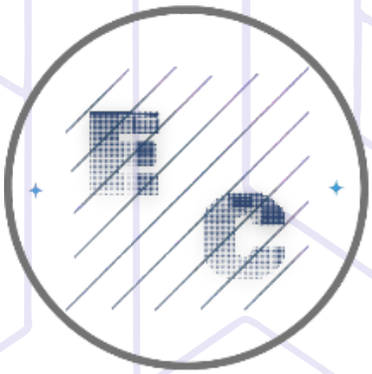
UNIVERSIDAD DE SONSONATE
USO
SCIENTIAE ET BONIS ARTIBUS



MULTIDISCIPLINARY JOURNAL EPISTEMOLOGY OF THE SCIENCES

Universidad de Sonsonate

Volume 3, Issue 3

July–September 2026
Special Edition

CROSSREF PREFIX DOI: 10.71112

ISSN: 3061-7812, www.omniscens.com

Multidisciplinary Journal Epistemology of the Sciences

Volume 3, Issue 3
July–September 2026, Special Edition

Quarterly Publication
Made in Mexico

The Multidisciplinary Journal Epistemology of the Sciences accepts submissions from any field of knowledge, promoting an inclusive platform for the discussion and analysis of epistemological foundations across various disciplines. The journal invites researchers and professionals from fields such as the natural sciences, social sciences, humanities, technology, and health sciences, among others, to contribute with original articles, reviews, case studies, and theoretical essays. With its multidisciplinary approach, it aims to foster dialogue and reflection on the methodologies, theories, and practices that underpin the advancement of scientific knowledge in all areas.

Contact: admin@omniscens.com

9773061781003

Legal Disclaimer

Multidisciplinary Journal Epistemology of the Sciences Vol. 3, Issue 3, July-September 2026, Special Edition, is a quarterly publication edited by Dr. Moises Ake Uc, C. 51 #221 between 16B and 16C, Mérida, Yucatán, Mexico, C.P. 97144, Tel. 9993556027, Web: https://www.omniscens.com, admin@omniscens.com. Responsible Editor: Dr. Moises Ake Uc. Rights Reservation No. 04-2024-121717181700-102, ISSN: 3061-7812, both granted by the Instituto Nacional del Derecho de Autor (INDAUTOR). Responsible for the last update of this issue: Dr. Moises Ake Uc, last modification date: July 1, 2026.

Multidisciplinary Journal Epistemology of the Sciences

Volume 3, Issue 3, 2026, July–September, Special Edition

DOI: https://doi.org/10.71112/pahqyx95

THEORY AND PRACTICE OF AI-AUGMENTED SOFTWARE ENGINEERING IN EUROPE IN 2025

TEORÍA Y PRÁCTICA DE LA INGENIERÍA DE SOFTWARE AUMENTADA POR INTELIGENCIA ARTIFICIAL EN EUROPA EN 2025

Denis Svyatoslavovich Pashchenko

Russian Federation

DOI: https://doi.org/10.71112/pahqyx95

Theory and practice of ai-augmented software engineering in Europe in 2025

Teoría y práctica de la ingeniería de software aumentada por inteligencia artificial en Europa en 2025

Denis Svyatoslavovich Pashchenko[a,*]

denpas@rambler.ru

https://orcid.org/0000-0001-9089-8173

*Corresponding author: denpas@rambler.ru, [a]Independent consultant and researcher, In software technologies, Moscow, Russian Federation.

ABSTRACT

This article examines the practical application of artificial intelligence tools, specifically ChatGPT versions 3.5 and 4, throughout the complete lifecycle of a real-world software development project. The study is based on field research conducted in Europe (2022-2023) and evaluates the effectiveness of AI in generating key software engineering artifacts, including functional specifications, source code, automated test cases, user documentation, and application content. The findings indicate that ChatGPT served as a highly effective assistant in the creation and refinement of these deliverables. However, its limitations became apparent in tasks requiring current technical knowledge, adaptation to rapidly evolving technologies, or the development of innovative business logic. Overall, the use of ChatGPT substantially accelerated the software development process, with project team members reporting significant improvements in productivity. The study further identifies best practices for integrating AI into software engineering workflows, emphasizing the importance of service-oriented architecture, iterative prompt engineering, and continuous human oversight. The results support the research

hypothesis that AI can reliably generate essential software development artifacts with minimal human intervention, thereby representing a significant step toward the broader adoption of AI-assisted software engineering practices.

Keywords: AI; software engineering; LLM; Android; ChatGPT.

RESUMEN

Este artículo explora la aplicación práctica de herramientas de IA (ChatGPT v3.5 y 4) en todo el ciclo de vida de un proyecto real de desarrollo de software, basado en investigación de campo realizada en Europa durante 2023–2024. La investigación evalúa la efectividad de la IA en la generación de artefactos centrales de software, incluyendo especificaciones funcionales, código fuente, pruebas automatizadas, documentación de usuario y contenido de aplicaciones. Si bien la herramienta de IA demostró ser un asistente potente para crear y refinar estos entregables, sus limitaciones se hicieron evidentes en áreas que requieren orientación técnica actualizada o aportes creativos en el desarrollo de la lógica de negocio del producto. En términos generales, ChatGPT aceleró significativamente el proceso de desarrollo, y los miembros del equipo del proyecto reportaron un aumento sustancial en la productividad. El estudio también destaca buenas prácticas para el uso de IA en ingeniería de software, enfatizando la importancia del diseño orientado a servicios, el refinamiento iterativo de prompts y la supervisión humana continua. A pesar de sus limitaciones en la generación de ideas originales o en la adaptación a requisitos cambiantes de plataformas, ChatGPT demostró sólidas capacidades para automatizar tareas repetitivas y mejorar la eficiencia global.
Los hallazgos confirman la hipótesis de investigación de que la IA puede producir de manera confiable artefactos clave del desarrollo de software con mínima intervención humana, marcando un paso decisivo hacia una integración más amplia de la IA en las prácticas de ingeniería de software.

DOI: https://doi.org/10.71112/pahqyx95

Palabras clave: AI; ingeniería de software; LLM; Android; ChatGPT.

February 16, 2026 | Accepted: June 30, 2026 | Published: July 1, 2026

INTRODUCTION

The development of software using artificial intelligence (AI) tools has remained a significant subject of both scientific inquiry and industrial practice for more than fifteen years. Throughout this period, researchers, technology companies, and software engineering practitioners have continuously investigated the potential of AI to transform the software development lifecycle by automating complex engineering activities, improving software quality, and increasing developer productivity. Numerous specialized and non-specialized organizations have attempted to forecast the future evolution of software engineering, consistently identifying AI technologies as one of the principal drivers of innovation and digital transformation (Gorban & Grechuck, 2018). These studies have envisioned software development environments in which intelligent systems actively support or automate a broad spectrum of engineering tasks, resulting in highly efficient production processes characterized by extensive automation, data-driven decision-making, and the widespread adoption of "intelligent" technologies (Kästner & Kang, 202).

Despite these long-standing expectations, the practical realization of AI-assisted software engineering remained limited for many years. Earlier generations of AI-based development tools typically exhibited constrained functionality, required substantial technical expertise, or were applicable only to narrowly defined programming tasks. Consequently, although the theoretical potential of AI in software engineering was widely acknowledged (Barenkamp et. al., 2020), its adoption in everyday software development practices remained relatively modest. AI technologies were primarily employed in isolated activities, such as static code analysis, defect prediction, automated testing, or recommendation systems, rather than as comprehensive assistants capable of supporting the entire software development lifecycle.

A fundamental turning point occurred in the autumn of 2022 with the public release of large language models (LLMs), particularly ChatGPT. For the first time, highly capable generative AI systems became readily accessible to millions of software developers, analysts, testers, technical writers, and project managers worldwide. This unprecedented accessibility, combined with significant advances in natural language processing, reasoning capabilities, and code generation, enabled a profound transformation in the way software projects could be designed, implemented, tested, documented, and maintained. AI evolved from being a specialized auxiliary technology into a general-purpose engineering assistant capable of participating in virtually every phase of software development.

Since that milestone, software engineering has undergone rapid organizational and technological change. Development teams have increasingly integrated AI tools into their daily workflows to support requirements elicitation, functional specification development, software architecture design, source code generation, automated test creation, technical documentation, user support materials, and various project management activities. Consequently, the emergence of modern generative AI has not merely introduced another software development tool but has fundamentally altered the organizational and production paradigms of the software engineering industry. The combination of advanced language models, widespread accessibility, and practical usability has enabled a level of automation that had previously been anticipated primarily in theoretical discussions.

Traditional twentieth-century software development centers, typically structured around project-based teams managed by local leadership, are increasingly being replaced by geographically distributed micro-teams (Boyko & Holoborodko, 2024). Within these new configurations, each core production function (e.g., systems analysis, project management, software design and development) is led by a domain expert fully equipped with automation tools capable of handling all routine tasks. It is important to note that the transition from in-office

to hybrid or fully remote work formats, a trend gaining significant momentum since 2020, has also played a critical role in this paradigm shift (Pashchenko, 2024). These new work modalities have profoundly altered established principles of team formation, hiring, and termination practices in the IT sector.

Concurrently, the role of AI technologies underpinning the concept of AI-augmented software engineering (Panetta, 2023) has become increasingly evident and transformative within this evolving context. The ability of IT companies to successfully adapt to these rapidly changing conditions has become essential for maintaining competitiveness (Cakmak, 2023). While this shift in the organizational and production paradigm has already begun, investment in its core trajectories demands a careful and balanced approach, thus necessitating further applied research.

This study sets out to conduct a fundamental assessment of the potential of AI tools to support the execution of a practical software development project from its earliest stages such as the initial idea or concept - through to the product's market release and consumer adoption. The research hypothesis posits that all critical artifacts of a software product including the technical specification, source code, and user documentation can be generated by AI tools with a sufficient level of quality and minimal labor input from individual engineers within the project team. Accordingly, the hypothesis assumes that the selected AI tools and the associated practices employed by the team have reached a level of operational maturity that allows them to function as comprehensive automation instruments for the everyday tasks of virtually every project team member. The proposed research problem aims to demonstrate the real-world capabilities and practical applications of AI tools in the core production functions of a software development project.

To ensure an adequate user experience during the implementation and utilization of AI technologies in software engineering, the findings of the author's earlier scientific study, "Early

Formalization of Large Language Model Utilization in Software Development" (mid-2023), were thoroughly analyzed (Pashchenko, 2023). Additionally, insights from a subsequent author-led study, "AI Tools in Software Production – Demands and Barriers," completed in the end of 2024 (Pashchenko, 2025), were employed to refine AI-assisted software development practices in 2025.

Both studies encompassed 60+ teams from IT companies, system integrators, and banks with robust in-house software development capabilities, spanning geographical regions from Russia and Kazakhstan to the United Kingdom and Spain. The teams represented diverse segments of the software development industry, including:

- Independent software vendors, including in-house and product development (e.g., Miro, Google, Finastra, Finshape, Sber, VTB, Playrix, OZON, PSB, Deutsche Bank);
- Custom software development and outsourcing firms (e.g., Atos, SOFTEC, First Line Software, and EPAM); System integrators (e.g., ThoughtWorks and Auxo);
- Other IT companies with complex economic models (e.g., Capgemini Engineering, ARM, ZEISS Digital, and Ericsson).

Both studies employed a mixed-methods approach, utilizing Google Forms for structured surveys alongside video interviews. The structured findings were subsequently distributed to domain experts, allowing them to provide commentary and contribute to the final interpretations of the research. The extensive panel of experts – 35 software development teams (Pashchenko, 2023), together with the wide geographic representation, provides a solid foundation for asserting that the resulting conclusions reflect at least the advanced practices currently prevalent in Europe (see Table 1).

DOI: https://doi.org/10.71112/pahqyx95

Table 1.

Experts in theoretical studies

No.	The characteristic	Representation in the expert's study		
1	Level of professional experience in software engineering	Less than 10 years	10-15 years	More than 15 years
	Kästner & Kang, 2020	22 %	19%	59%
	Gorban et al., 2018	11%	22%	67%
2	Region of described the experience in usage of AI-tools	North and West. Europe (Spain, Sweden, France, Germany, Swiss, UK, etc)	Central and South Europe (Poland, Czechia, Hungary, Serbia Bulgaria, Cyprus, etc)	Eastern Europe and CIS (Ukraine, Russia, Armenia, Turkey Kazakhstan, Georgia, etc)
	Kästner & Kang, 2020	11%	33%	56%
	Gorban et al., 2018	29%	20%	51%
3	Types of IT-business	Independent software vendors, including in-house product development	Custom development and outsourcing software services	Other type of IT business
	Kästner & Kang, 2020	56%	33%	11%
	Gorban et al., 2018	46%	29%	25%

Principal results and key findings of those studies (Pashchenko, 2025) had been used in practical implementation of AI-augmented software development paradigm in 2023-2025 in Android development teams of software company in Spain. Summary of the study's results and result of AI implementation (AI-Augmented Software Engineering paradigm) are given in following sections of the article.

DOI: https://doi.org/10.71112/pahqyx95

METHODOLOGY

This study shows how theoretical result are forming the frame for practical implementation of AI-Augmented Software Engineering paradigm in real software industry. The goal of the study is defining the best practices of using AI instruments in real project of product's software development. Those best practices had been defined in mentioned above research and made a solid base for practical implementation. The methodology of the research is based on the following provisions:

1. Decomposition, system analysis, synthesis - for processing the results of theoretical research and collecting promising methods and approaches for implementing innovation;
2. Change management and project management for implementing AI tools in software development processes;
3. Empirical assessment and system analysis of production indicators for analyzing intermediate and final results of implementing innovation.

The description of the process of the implementation of AI tools in the practice of implementing a software project requires delving into the details of the study:

1. Timeframe: The software development project was executed from July 2023 to April 2025, and involved a geographically distributed production team of five people from the IT company Slavasoft. The development was carried out in the Scrum production paradigm (Misra et al., 2012).
2. Software product: Android-app for smartwatch branded as «AI Alter Ego», implementing the functions of a digital personal assistant in a smartwatch. During the project, three major releases of the system were released. Three releases of the system in 2024-2025 fully implemented all the intended functionality of the system for the Google Wear 3, 4 and 5 operating system installed on smartwatches of all

leading global manufacturers (except Apple) (Barai & Mutreja, 2023). The Alter Ego software product is available for download from the Google.Market app store for free and without geographic restrictions.

3. Artifacts which are necessary for the release of this software product:
    - System vision (business requirements level),
    - Functional specifications,
    - Architectural diagrams, such as component and Archimate-TOGAF (Wierda, 2021),
    - Software code,
    - Test plan with test cases,
    - User and service documentation for the software product.
4. As an AI tool that implements the tasks set for the development team, the ChatGPT service version 3.5 (2023) and version 4.0 (2024-2025) from OpenAI, which has gained enormous worldwide popularity since the fall of 2022 (Ahlgren, 2023).

Thus, this study sets a relevant scientific task of fundamentally assessing the capabilities of an engineering team to create all key artifacts of a software project using an artificial intelligence tool that is available on a permanent free basis. The results of two applied studies were used as a theoretical basis necessary for managing the implementation of AI tools. A brief summary of these results is given in the next section of the article.

RESULTS

By the end of 2024 there were clearly observed: innovators, early adopters, early majority, late majority, and laggards (Rogers, 2003). An interesting observation is that the introduction of AI tools into the actual practice of software production follows this classification:

- Categories with "innovators" and "early adopters" according to E. Rogers were formed and the formalization of the use of AI tools began (project teams carry out a corporate plan for introducing AI into software development, create and use centralized corporate policies and/or recommendations);
- The process of forming the category of "early majority" according to E. Rogers is finished to 2025 - teams finished the studying AI tools in various ways (R&D, individual / team experiment, etc.) and started its implementation in software development.

Around 23% of experts are using AI tools in their regular job with the high frequency and it has a strong impact on their personal tasks in IT projects. Then in the end of 2024 the study shown the rising of the share (to 35%) of teams and organizations already has started the implementation of AI-tools in real software production (in different process areas like coding, testing, etc) and it's planned to do in near future for 48% of teams. Development in the real practice allows:

- Automation of the routine operations and time saving;
- Speed up of the operations in the team \ organization;
- Software product excellence, including software quality, UX and documentation.
- Also both studies had shown that AI-tools are not in high demand for business and system analysis, but there were much more popular types of tasks for AI-tools: coding, code reviews, fast software prototyping.

About 63% of experts in study used the AI-tools in making software code and were satisfied with the reached result. Both studies demonstrated that impact of AI-tools on software quality management is rising and there were defined most popular types of tasks for LLMs in software quality management: writing the auto-tests, managing the defects and reports

analysis;searching of the errors and vulnerabilities in the code. But still around the half of experts in study (Pashchenko, 2025) does not use AI-tool in software quality management.

Expert panel estimated the value and the role of LLMs in learning and in the excellence of the software development skills in the end of 2024:

- The impact of usage LLMs in professional learning is very high – 41% of experts;
- It's just one more useful tool on the board – 44% of experts.

For sure, the implementation of LLMs has their specific features and risks, that we need to estimate before the implementation of AI-tools in software production. Expert's panel in study figured the main barriers in the implementation of any AI-instruments in the software development in their teams and organizations:

- High level of different risks - from legal aspects to ethical – 44% of experts;
- Lack of the resources (money, time, knowledge, HR-capital) – 30% of experts;
- Organizational resistance of engineers and managers – 18% of experts.

But those risks are not a solid barrier – more and more IT companies are in the active process of AI-tools implementation in the software engineering practices. From 12% of teams in study to 26% of teams in study are executing an official plan in how to use AI-instruments in IT projects, and in around 30% of teams its usage is continuing in test mode in their teams without centralized management.

So, study (Pashchenko, 2025) confirmed the earlier results from:

- Innovator's and early adopter's groups are formed in software development domain; they had started the AI-tools implementation in software production.
- Current AI-instruments usage is focused on the working with the software code (in different kind of ways), on the quality assurance and on the tasks about documentation.

- For some IT companies the lack of centralized efforts on the corporate level might lead to the missing of the competition advantage in the software production, connected with AI tools.
- Software engineers need new skills in AI-human interaction: as earlier company will start educate them as more effective they will be in future production process.

This leads to some considerations about best practices in AI tools implementation. First, focus AI usage in software engineering on main process areas in the production, start with all activities where software code and product documentations are the expected result. Second, formalization of the process of the implementation of AI into software engineering is a key factor in overcoming all barriers. And third, rising of interest to AI from industry and governments lead to the need of complex risk management in internal projects of implementation of AI.

Moreover, the above results have identified a set of practical questions for the current study, which are formulated in Table 2. This set of questions allows us to solve the scientific problem posed above and test the proposed research hypothesis for the full software life cycle: from the earliest stages (idea, concept) to business and system requirements collected in the technical task, to system design and creation of program code, then to product implementation and to its release into industrial environments accessible to end users.

Table 2.

Areas of software development and corresponding questions

№	Software Engineering Process Area	Current Research Question
1	System Analysis	How can an AI tool be practically useful in creating a technical specification at all stages: ¿from decomposition and analysis of requirements to its documentation?

2	Software Design and Construction	How can an AI tool be practically useful in designing a product architecture, quickly prototyping functions and creating stable working software code? How quickly does an AI tool fail when the logic is significantly complicated and there is a need to support the related software code of several modules or services?
3	Software Quality Management	How can an AI tool be practically useful in software quality management: ¿from creating automated tests to generating test cases for manual testing?
4	Software Content Creation	¿Can an AI tool create diverse content, high-quality and expert in the subject area of automation?
5	Software Documentation	Can an AI tool create detailed and accurate project documentation based on software code and requirements - user manuals, service documents, ¿etc.?

Also, to solve the scientific problem of this study, it is necessary to take into account the results of studies, namely, what difficulties and barriers exist in scenarios for the implementation of AI tools in the activities of a software development company (Sattelberg, 2023; Glinka & Raed, 2009). Of course, the implementation of AI tools has its own characteristics and risks that must be managed on an ongoing basis.

Practical case of AI implementation in software engineering

This section of the study, focused on the practical case of AI tool implementation, details the process and outcomes of the standardized use of ChatGPT 3.5 (in 2023) and 4.0 (in 2024) within a software development workflow based on the Scrum methodology. The software company SlavaSoft initiated the development of the “Alter Ego” application for Android-based smartwatches from scratch, beginning with a product "vision" document. Since late 2023, a dedicated team of five engineers has employed the AI tool to design software and generate all related project documentation.

The adoption of AI tools within the company began with this team and, starting in 2023, all new innovation initiatives have been structured as “pilot projects”. The implementation was guided by an official change management plan and aligned with the company’s product roadmap. Even prior to the project launch, the team conducted several meetings to discuss AI tool usage for software coding and testing in light of previous research findings (Pashchenko, 2023) y (Pashchenko, 2025), with the objective of developing internal best practices. In the end of 2023 were built official policy – how to use AI tools for solving typical tasks in software production processes, including code, tests and documents generation.

Best practices in particular software project are inherently dependent on the type of software product under development. Accordingly, understanding the complexity and associated risks of the project, as well as estimating development and quality assurance efforts, is critical. “Alter Ego” is a sophisticated Android application with over 50 features, designed to deliver immediate AI-based care and assistance via a smartwatch interface. As stated in the product specification: “Alter Ego” is the first deeply human-centered AI assistant built exclusively for Wear OS smartwatches”. This positions “Alter Ego” as a feature-rich and complex application, with strong emphasis on smartwatch-specific usability and interface design.

It is important to note that the selection of ChatGPT as the primary AI tool for this project followed a careful evaluation process. The main considerations included:

- The availability of a free subscription account enabling full-day work on code and documentation generation (for version 3.5);
- The rapid advancement of the tool, along with its versatility across nearly all tasks associated with software development.

Other AI tools, such as Microsoft Copilot (in 2023) and Cursor (in 2025), which are integrated into software development environments, were also considered. However, the

aforementioned advantages of ChatGPT proved more compelling for the project team than factors such as ease of integration or response speed.

It should also be emphasized that ChatGPT was not integrated into the development environment (e.g., Android Studio for code or Atlassian Jira for task and requirements management). All interactions with the AI tool occurred within its native interface, with dialogues stored on the platform's side. Key outputs were manually transferred into project artifacts.

The general algorithm for AI tool usage was consistent across tasks and included the following steps:

1. Construction of a precise and detailed prompt describing the task in terms of system requirements;
2. Verification of the response and, where necessary, refinement through iterative prompting in dialogue mode;
3. For code generation tasks, prompts were often decomposed to ensure each response was limited to 300–400 lines of code in 2023 and 600-700 lines of code in 2025, supporting consistency and logical coherence throughout the engineer-AI interaction;
4. In some instances, entire AI-generated responses were re-submitted as input for further refinement or correction.

Once verified and finalized, AI-generated outputs were incorporated into project artifacts such as technical specifications, working system code, user documentation, and test cases. For software code in Java and XML, the finalized outputs were integrated into the corresponding modules of the Android project within the Android Studio environment. Over the course of development, both the codebase and the software service structure evolved. Nevertheless, by 2025, only two of the more than 50 system services (specifically those comprising the core business logic) did not contain any AI-generated code.

One notable limitation identified was the AI tool's restricted ability to maintain logical coherence across large codebases. Empirical observations indicated that version 4.0 could successfully interpret and manage interdependencies across services sharing common data, with a practical upper limit of approximately 700 lines of code per response. Attempts to generate abstract-level inter-service logic (i.e., based on semantic rather than structural or data relationships) consistently failed to meet the project team's quality expectations. It is also relevant that the team did not utilize collaborative AI tools. Each engineer independently interacted with the AI tool on a prompt-response basis. However, the team regularly reviewed and discussed generated artifacts, especially documentation, and to a lesser extent, source code. The outputs from ChatGPT were manually processed and refined by engineers and served as the foundation for all major project artifacts, including technical specifications, program code (Java, XML), test cases, and product documentation.

Simultaneously, certain critical artifacts - such as the business vision document (defining high-level business requirements) and architectural diagrams (covering component, application, and data views in accordance with the ArchiMate (TOGAF) framework (Wierda, 2021) - were created exclusively by the engineering team, without the usage of AI tools.

Although the initial technical specification was fully aligned with the business vision at the project's inception, new ideas and requirements emerged throughout the development cycle. These were rapidly prototyped and incrementally incorporated into the evolving technical documentation. This approach aligns with established industry practices wherein requirements management continues throughout the software development lifecycle (Sommerville, 2009).

The overall results of using ChatGPT in the project are shown in Table 3 (collected from project documentation and interviews with engineers):

Table 3

AI-tools usage findings from project Alter Ego

DOI: https://doi.org/10.71112/pahqyx95

№	Software Engineering Process Area	Current Study Question (from Table 2)	ChatGPT 3.5 Usage Results (2023-2024)	ChatGPT 4 Usage Results (2024-2025)
1	System analysis	How can an AI tool be practically useful in creating a technical specification at all stages: from decomposition and analysis of requirements to its documentation?	Exceeds team expectations: AI creates a technical task of any complexity, based on business requirements from the project team. The result: a structured document with a sufficient level of nesting and detail for the practical development of a software product.	Exceeds team expectations: AI creates a technical task of any complexity based on business requirements from the project team. The result: a structured document with a sufficient level of nesting and detail. AI itself suggests additional sections and finds contradictions in system requirements.
2	Software product design and construction	How can an AI tool be practically useful in designing a product architecture, quickly prototyping functions and creating stable working software code? How quickly does an AI tool fail when the logic is significantly complicated and there is a need to support the related software code of several modules or services?	Architecture design – below the team's expectations, since the tool does not support graphical visualization of the component diagram, however, at the text level, it describes the composition of classes and their interaction with each other and, for example, with the database - correctly and in detail. Rapid prototyping (under conditions of unclear or incomplete requirements) - above expectations, in about 80% of cases, the AI tool built a correct prototype of the future function at the	Architecture design – below the team's expectations, still no visualization support. At the text level, it decomposes in detail and in a coherent manner, selects technological priorities for implementation. Rapid prototyping (under conditions of unclear or incomplete requirements) – above expectations, in about 95% of cases, the AI tool built a correct prototype of the future function at the program code level on the first try.

			program code level on the first try. Working versions of the program code - meets the team's expectations: the more detailed the requirements for the code were described, the more accurate and logically consistent the result was. All code was verified multiple times in the project, both by engineers and with the help of ChatGPT. The ChatGPT tool supports the logical structure and consistency of the program code only for closely related services (integrated) and if all the service code is loaded into the dialog with the tool.	Working versions of the program code – meets the team's expectations: the more detailed the requirements for the code were described, the more accurate and logically consistent the result was. All code was verified multiple times in the project, both by engineers and with the help of ChatGPT. As before, the ChatGPT tool supports the logical structure and consistency of the program code only for closely related services (integrated) and if all the service code is loaded into the dialog with the tool.
3	Software quality management	How can an AI tool be practically useful in software quality management: from creating automated tests to generating test cases for manual testing?	The team had a controversial impression of the automated tests; in some cases, the automated test logic was not implemented in the best way even with multiple adjustments. Test cases for manual testing – meet the team's expectations, although a specialist in the team manually significantly	Autotests - meet the team's expectations, are created correctly and logically consistent. Test cases for manual testing - meet the team's expectations, although a specialist in the team still manually significantly supplemented them to the working version.

			supplemented them to the working version.	
4	Software content creation	Can an AI tool create diverse content, high-quality and expert in the subject area of automation?	The content of the application (expert data on the field of automation, lists of standard values, dictionaries, etc.) is beyond any expectations. When formulating precise queries, the content was created in any required volume while maintaining high data quality.	Exceeds all expectations of the team. The emergence of the visual content generation function further expanded the capabilities of the development team in creating the application. When formulating precise requests, content was created in the required volume while maintaining high data quality.
5	Software documentation	Can an AI tool create detailed and accurate project documentation based on software code and requirements - user manuals, service documents, etc.?	User and service documents - meet the team's expectations. Moreover, the AI tool was equally good at creating voluminous instructions for users and short service documents on ready-made java code.	Any user documents - exceed any team expectations. AI itself suggests missing sections and corrects the document structure.

It is also worth describing the use of the AI tool for other tasks that turned out to be relevant for this software development project. Thus, when significant engineering problems arose for the development and testing environments (including those involving real smartwatches from various manufacturers (Chuah et. al., 2016)), the team turned to ChartGPT for technical advice. However, due to the limitation of this tool in having the most up-to-date information, all the answers were template-based and significantly inferior to the team's current knowledge. The experience of consulting on placing an application in the Google Play Market store was also unsuccessful: Google's tightening of policies on placing and updating applications for smartwatches since 2021

made the publication process itself more complex (Sattelberg, 2023). Attempts to find a solution to constantly emerging operational issues using the AI tool were also unsuccessful - the data inside ChatGPT turned out to be outdated and irrelevant in this area.

Thus, the capabilities of AI tools (using ChatGPT 3.5 and 4 as an example) in implementing a practical software development project should be assessed as sufficient. The research hypothesis has been confirmed: all significant software product artifacts – functional specifications, software code, user documentation in a real software development project can be generated by AI tools with minimal effort from the relevant engineers in the project team. This means that the scientific task has been solved, and the AI tool chosen by the team - ChatGPT from OpenAI has already reached the required high level of efficiency for the fundamental tasks of software engineering and is an effective tool for automating project activities.

In the end of the description of the results of the implementation of AI tools, several conclusions should be made about their most effective usage. These provisions were formulated from the results of retrospectives (meetings of engineers to discuss the work done and the problems solved) after development sprints and they are categorized by areas of application in software engineering.

1) Creating a technical specification based on business-level requirements (e.g. based on a business vision document) is most effectively done for system services, which implies the need to identify individual services during design at the earliest design stages. The ChatGPT tool generates system and technical (non-functional) requirements much more effectively if a service oriented architecture is applied (Glinka & Raed, 2009) and the business sense of its operation is specified for each service (e.g. in the form of value for the end user).

2) Generating system code (both during rapid prototyping and for the working version after analyzing and coordination of the requirements) is an iterative process, it’s aligned with other research (Ridi, 2024). Persistence in clarifying prompt requests and high structuring and detailing

of system requirements are of decisive importance. In some cases, the ChatGPT tool behaves non-deterministically: sometimes it misses some of the requirements, sometimes it “invents” additional business logic. Persistence in repeating precise and detailed requests leads to success in 95% of cases, even if the first answer of the AI tool was completely inaccurate. The approach of sending the generated code back to the ChatGPT dialog with additional instructions worked well; within 1-3 additional iterations, the tool produced fully working code that met the initial system requirements.

3) Best working prompts for code and documents generation had been done in formal structure:

a. **Role and Context** (for AI agent in task of prompt),

b. **Step-by-step algorithm** (how AI should do the task);

c. **Output format** (language, format, structure, etc);

d. **Final instruction** (additional remarks, including level of creativity for AI).

4) Auto-tests and test cases for manual testing as part of ensuring high quality of the software product were generated based on the existing system code. It’s aligned with modern research (Peng et. al., 2023) and (Petrovic et. al., 2024)), but not very common in software development. Of course, these artifacts were verified and refined (expanded and clarified) by the software quality assurance engineer, but in his opinion, the results could have been immediately used in the project if there had been such a production need. This is an obvious paradox, since the purpose of testing in software engineering is to check the code for compliance with requirements, but the team managed to obtain good quality testing artifacts immediately based on the code and even identify code sections (without testing) that had previously been implemented not quite accurately. This conclusion is in deep contradiction with traditional approaches to software quality assurance and requires further reflection: if valid (in the opinion of

the team) test artifacts are obtained based on the code (and not on system requirements and the work of the corresponding engineer), can they help to find defects at all? According to the author, the reason for this paradox in a specific project is the high atomicity of test cases and the simplicity of the business logic of the software product.

5) User documentation for the already generated code of the working system was completely created in ChatGPT in 2024. The result of the work was so good that the technical writer involved in the team part-time only had to add illustrations (screenshots of the system by functions).

6) Creating application content using AI using the Alter Ego software product as an example turned out to be the fastest of all existing methods. The team did not need any help from external specialists to create 100% of the application content in the AI tool. The emergence of the ability to generate visual objects in ChatGPT version 4 also made it possible to automate some of the marketing functions for the application (marketing leaflets, graphics for the website). It's fully aligned with results of studies (Pashchenko, 2023) and (Naimi, 2024) ;

7) Creating a software product is the implementation of a certain business logic (and not just a set of loosely coupled application functions). ChatGPT understands the most complex logic (both at the level of the requirements text and at the level of software code) and can offer its improvement according to the specified requirements of the project team as it was defined earlier in (Kurnianingrum, 2024). At the same time, the generation of new ideas for AI tools in terms of business logic is template-based and clearly uncreative, i.e. seriously inferior to the capabilities of any specialist in the field of software engineering. The team doubts that the tool used could create the original business logic of the new software product without serious and significant human involvement. Those doubts in practical project are aligned with results of study (Shiye & Chien-Ming 2022).

8) No one in the project team had any doubts that the selected AI tool allows performing personal tasks in the areas of software engineering much faster and more efficiently than without this automation. There was no practical sense for the project team to calculate the percentage of increasing efficiency or economic feasibility, because after only two months of using the selected tool, the team's work had changed significantly: not a single member of the project team wanted to work on the project without this tool anymore.

CONCLUSIONS

Applied research has identified the implementation features and key aspects of using artificial intelligence tools in the practice of software development companies. Research in 2023-2024 have shown that a significant proportion of organizations have already begun implementing AI tools in real software production. The use of AI in software development automates typical and routine operations and is in demand due to the need to speed up software development and improve the quality of the final product. Although experts from (Pashchenko, 2023), (Pashchenko, 2025) noted that by the end of 2024, AI tools were not in great demand for solving business and system analysis problems, they were nevertheless in significant demand in writing working code, rapid software prototyping, ensuring software quality management and some other areas.

As part of this study, the scientific problem of determining the fundamental capabilities of AI tools in creating key artifacts of a software project at all stages of its life cycle was solved - from technical specifications to working code and accurate user documentation for the software product. The process of using the AI tool by the development team was algorithmic and standardized in corporate policy: from formulating precise requests to the selected ChatGPT tool to applying the received and verified answers in practice. The development team engineers

worked with the AI tool individually, but the final results were regularly discussed and processed by the entire team.

The ChatGPT tool demonstrated high efficiency in creating technical specifications, rapid prototyping of functions and generating software code. Importantly, requirements management was continued throughout the project, which allowed for flexible adaptation to new ideas and constant (albeit small) changes in the area of business requirements. The AI tool was used for engineering consultations, creating technical specifications, generating system code, developing automated tests and test cases, compiling user documentation and creating application content. As a result, the following conclusions were made:

- Using AI to create technical specifications and user documentation proved to be effective, especially when working with service architecture and detailed description of business value for the user when using each service;
- Generating system code using AI was successful, but it was iterative, in which persistence in interaction with AI is important (constant repetition of context and requirements management in prompts). There are noticeable limitations in "remembering" by the AI tool of the logical connectivity of individual services in the system;
- AI-powered generation of application content was fast and efficient, requiring no additional assistance from external specialists. At the same time, AI's capabilities in creating the product's business logic were limited and required significant human intervention to generate original ideas.
- Overall, the use of ChatGPT met the project team's expectations in most respects, although there were moments that required additional attention, intellectual effort, and adjustments. Based on the results obtained, it should also be concluded that the

use of AI in software product development is an effective tool, although it requires the use of certain skills, control, and regular verification of results by a software engineer.

Conflict of interest declaration

The author declares no conflict of interest for this research.

Author Contributions Statement

Denis Svyatoslavovich Pashchenko: Conceptualization, writing – original draft.

AI use declaration

The author declares that artificial intelligence was used as a support tool for this article and for data analysis, given the nature of the topic. However, this tool does not in any way replace the intellectual task or process. After rigorous reviews using different tools, which verified that no plagiarism exists as evidenced in the supporting documentation, the author states and acknowledges that this work is the result of his own intellectual effort and has not been written or published on any electronic or AI platform.

REFERENCES

Gorban, A., Grechuk, B., & Tyukin, I. (2018). *Augmented artificial intelligence: A conceptual framework*. *arXiv*. arXiv:1802.02172 [cs.AI].

Kästner, C., & Kang, E. (2020). Teaching software engineering for AI-enabled systems. En *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training* (pp. 45–48). https://doi.org/10.1145/3377814.3381714

Barenkamp, M., Rebstadt, J., & Thomas, O. (2020). Applications of AI in classical software engineering. *AI Perspectives, 2*, 1. https://doi.org/10.1186/s42467-020-00005-4

Boyko, O., & Holoborodko, Y. (2024). *Distributed software development in 2025: All you should know*. SPD Tech Blog. https://spd.tech/dedicated-development-teams/distributed-software-development-team/

Pashchenko, D. S. (2024). Consolidation of a new organizational and production paradigm in software development projects. *Information Technologies*, (3), 150–158. https://doi.org/10.17587/it.30.150-158

Panetta, K. (2023). *Set up now for AI to augment software development*. Gartner Report. https://www.gartner.com/en/articles/set-up-now-for-ai-to-augment-software-development

Cakmak, Z. (2023). Adapting to environmental change: The importance of organizational agility in the business landscape. *Florya Chronicles of Political Economy, 9*, 42–53. https://doi.org/10.17932/IAU.FCPE.2015.010/fcpe_v09i1004

Pashchenko, D. S. (2023). Early formalization of AI-tools usage in software engineering in Europe: Study of 2023. *International Journal of Information Technology and Computer Science, 15*(6), 29–36. https://doi.org/10.5815/ijitcs.2023.06.03

Pashchenko, D. S. (2025). Crecimiento en la demanda de herramientas de inteligencia artificial en ingeniería de software: Resultados de un estudio paneuropeo 2024. *Revista de Investigación en Tecnologías de la Información, 13*(29), 82–91. https://doi.org/10.36825/RITI.13.29.008

Misra, S. C., Kumar, V., Kumar, U., Fantazy, K. A., & Akhter, M. (2012). Agile software development practices: Evolution, principles, and criticisms. *International Journal of Quality & Reliability Management, 29*, 972–980.

Barai, T. V., & Mutreja, S. (2023). *Smartwatch market: Global opportunity analysis and industry forecast, 2023–2032*. Allied Market Research. https://www.alliedmarketresearch.com/smartwatch-market

Ahlgren, M. (2023). *OpenAI statistics & facts for 2023 (DALL·E, ChatGPT & GPT-3.5).* https://www.websiterating.com/research/openai-statistics/

Rogers, E. (2003). *Diffusion of innovations* (5th ed.). Simon & Schuster.

Wierda, G. (2021). *Mastering ArchiMate Edition 3.1: A serious introduction to the ArchiMate enterprise architecture modeling language*. R&A.

Sommerville, I. (2009). *Software engineering* (9th ed.). Addison-Wesley.

Chuah, S., Rauschnabel, P. A., et al. (2016). Wearable technologies: The role of usefulness and visibility in smartwatch adoption. *Computers in Human Behavior, 65*, 276–284. https://doi.org/10.1016/j.chb.2016.07.047

Sattelberg, W. (2023). *Google is updating app guidelines for smartwatches just in time for Wear OS*. Android Police. https://www.androidpolice.com/2021/09/03/google-is-updating-app-guidelines-for-smartwatches-just-in-time-for-wear-os-3/

Glinka, F., & Raed, A. (2009). A service-oriented interface for highly interactive distributed applications. En *European Conference on Parallel Processing* (Lecture Notes in Computer Science, Vol. 6043, pp. 266–277). https://doi.org/10.1007/978-3-642-14122-5_31

Ferdiana, R. (2024). *The impact of artificial intelligence on programmer productivity*. En *International Conference on Software Engineering and Information Technology (ICOSEIT) 2024* (Vol. 2).

https://www.researchgate.net/publication/378962192_The_Impact_of_Artificial_Intelligence_on_Programmer_Productivity

Finio, M., & Downie, A. (2024). *La IA en el desarrollo de software*. IBM Review. https://www.ibm.com/es-es/think/topics/ai-in-software-development

Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). *The impact of AI on developer productivity: Evidence from GitHub Copilot*. https://arxiv.org/pdf/2302.06590.pdf

Petrovic, N., Lebioda, K., Zolfaghari, V., Schamschurko, A., Kirchner, S., & Purschke, N. (2024). LLM-driven testing for autonomous driving scenarios. En *Proceedings of the 2024 2nd International Conference on Foundation and Large Language Models (FLLM)* (pp. 173–178).

Naimi, L., Bouziane, E., et al. (2024). Automating software documentation: Employing LLMs for precise use case description. *Procedia Computer Science, 246*, 1346–1354. https://doi.org/10.1016/j.procs.2024.09.568

Kurnianingrum, D., Jumbri, I., et al. (2024). Exploring the Chat GPT's impact and prospects for business research purposes. 1–6. https://doi.org/10.1109/INOCON60754.2024.10511483

Cao, S., & Huang, C.-M. (2022). Understanding user reliance on AI in assisted decision-making. *Proceedings of the ACM on Human-Computer Interaction, 6*(CSCW2), 1–23.