Real-World AI Systems
https://journals.cypedia.net/rwas

ARTICLE

Theory and Practice of AI-Augmented Software Development in Europe

Denis S. Pashchenko

Independent Researcher, Moscow 123000, Russia

ABSTRACT

This article explores the practical application of artificial intelligence (AI) tools (ChatGPT v3.5 and 4) in the full lifecycle of a real-world software development project, based on field research of 2023–2024 in Europe. The research evaluates the effectiveness of AI in generating core software artifacts, including functional specifications, software code, automated tests, user documentation, and application content. While the AI tool proved to be a powerful assistant for creating and refining these deliverables, its limitations became evident in areas requiring up-to-date technical guidance or creative input in product business logic development. In general, ChatGPT significantly accelerated the development process, with project team members reporting a substantial increase in productivity. The study also highlights best practices for AI usage in software engineering, emphasizing the importance of service-oriented design, iterative prompt refinement, and ongoing human oversight. Despite its shortcomings in generating original ideas or adapting to evolving platform requirements, ChatGPT demonstrated strong capabilities in automating repetitive tasks and enhancing overall efficiency. The findings confirm the research hypothesis that AI can reliably produce key software development artifacts with minimal human input, marking a pivotal step toward the broader integration of AI in software engineering practices.

Keywords: AI; Software Engineering; Large Language Model (LLM); Android; ChatGPT

*CORRESPONDING AUTHOR:
Denis S. Pashchenko, Independent Researcher, Moscow 123000, Russia; Email: denpas@rambler.ru

ARTICLE INFO
Received: 25 August 2025 | Revised: 10 October 2025 | Accepted: 19 October 2025 | Published Online: 26 October 2025
DOI: https://doi.org/10.64797/rwas.v1i2.104

CITATION
Pashchenko, D.S., 2025. Theory and Practice of AI-Augmented Software Development in Europe. Real-World AI Systems. 1(2): 33–44.
DOI: https://doi.org/10.64797/rwas.v1i2.104

1. Introduction

The development of software using artificial intelligence (AI) tools has remained a pertinent scientific and practical issue for over fifteen years. Numerous specialized and non-specialized organizations have sought to predict the future of software engineering, wherein AI tools occupy a prominent role [1,2], and to envision innovative production processes characterized by extensive automation and "intelligent" technologies [3]. However, it was not until the autumn of 2022 that the practical advancement of relevant AI tools—combined with their unprecedented accessibility to millions of developers worldwide—enabled a tangible shift in the organizational and production paradigms of the software development industry.

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 [4]. 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 [5]. These new work modalities have profoundly altered established principles of team formation, hiring, and termination practices in the Information Technology (IT) sector.

Concurrently, the role of AI technologies underpinning the concept of AI-augmented software engineering [6] 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 [7]. 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 [8]. Additionally, insights from a subsequent author-led study, "AI Tools in Software Production—Demands and Barriers," completed at the end of 2024 [9], 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 in Pashchenko [8] and 27 teams in Pashchenko [9],

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**).

Table 1. Experts in theoretical studies.

№	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
	Pashchenko (2025) [9]	22%	19%	59%
	Pashchenko (2023) [8]	11%	22%	67%
2	Region of described experience in the usage of AI tools	North and West. Europe (Spain, Sweden, France, Germany, Switzerland, UK, etc.)	Central and South Europe (Poland, Czechia, Hungary, Serbia, Bulgaria, Cyprus, etc.)	Eastern Europe and Commonwealth of Independent States (CIS) (Ukraine, Russia, Armenia, Turkey, Kazakhstan, Georgia, etc.)
	Pashchenko (2025) [9]	11%	33%	56%
	Pashchenko (2023) [8]	29%	20%	51%
3	Types of IT-business	Independent software vendors, including in-house product development	Custom development and outsourcing software services	Other types of IT business
	Pashchenko (2025) [9]	56%	33%	11%
	Pashchenko (2023) [8]	46%	29%	25%

Principal results and key findings of Pashchenko [8,9] were used in the practical implementation of an AI-augmented software development paradigm in 2023–2025 by 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 the following sections of the article.

2. Goal, Materials and Methods

This study shows how theoretical results are forming the frame for the practical implementation of the AI-Augmented Software Engineering paradigm in the 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 the research mentioned above 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 [10].
2) Software product: Android app for smartwatch branded as "AI Alter Ego", implementing the functions of a digital personal assistant in a smartwatch (AI Alter Ego. Smart digital assistant for Android smartwatch. URL: https://play.google.com/store/apps/details?id=slavasoft.alterego). During the project, three major releases of the system were released (the first was the minimum viable product, and the second—all the main functions—Major, the third—Ultimate—extended functions to improve the user experience in caring about their health and well-being). 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)[11]. 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 (component and Archimate—TOGAF—The Open Group Architecture Framework),
   - 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[12], was chosen.

This study methodologically represents a generalization of theoretical studies and verification of their results using a specific example of industrial software development. Custom development approach in analyzed case had some additional limits, as described below.

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.

Limits of the Study

1) The main results of the theoretical studies 2023–2024 were verified on the example of a single project of software development. The main assumption here is simple: if a scientific task might be resolved in this example with almost zero USD investments, then the study's results might be easily extrapolated to any projects where a software company is ready to invest money in this process.
2) In the analyzed custom development project, those teammates had worked together for the first (and last) time.
3) Research questions are aligned with the defined scientific task above. No practical value for industry in defining any quantitative metrics (like development velocity, defect density, review effort), because all those metrics are strictly dependent on a particular project (software product, tech stack, level of automation) and a particular team (persons, processes, motivation, etc.). In an analyzed practical case of software development, nobody will repeat the same product via the same team that makes any "numeric assumptions" out of economic sense.
4) The study does not have any control group of software developers, and nobody really cares about counting time or any other numeric parameters for developers, who do the same or similar tasks with and without AI tools.
5) The software development team had worked with different versions of the AI tool (conclusions are separated by those versions in the following tables); moreover, even during usage of the same version of GPTChat 3.5 or 4, those LLMs evolved very rapidly.

3. Results

3.1. Main Results of the Theoretical Studies 2023–2024

Both Pan-European studies[8,9] confirmed the rising demand for AI-augmented software development in the IT industry. Moreover, it fully corresponds with innovation diffusion, described by E. Rodgers in his well-known theory of innovations[13].

By the end of 2024, there were clearly observed: innovators, early adopters, early majority, late majority, and laggards. 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 by 2026—teams finished studying AI tools in various ways (Research and Development or R&D, individual/team experiment, etc.) and started their implementation in software development.

In the middle of 2023, Pashchenko[8] showed that a significant share (20%) of teams and organizations had already started the implementation of AI tools in real software production, and it was planned to do so in the near future for around 43% of teams. And personally, around 23% of experts are using AI tools in their regular job with high frequency, and it has a strong impact on their personal tasks in IT projects. Then, at the end of 2024, Pashchenko[9] showed the rising share (to 35%) of teams and organizations that have already started the implementation of AI tools in real software production (in different process areas like coding, testing, etc) and it's planned to do so in the near future for 48% of teams. Personally, in Pashchenko[9], around 37% of experts were using AI tools in their regular job with high frequency and it has a strong impact on their personal work in software development projects. And 30% of experts estimated the impact of AI tools on their professional work as average and valuable for some particular tasks in software development projects. It means that more than 2/3 of experts from Pashchenko[9] are using AI tools in software development projects regularly.

Both studies defined the main advantages of AI/LLMs usage in software development in real practice:

- Automation of the routine operations and time saving;
- Speed up the operations in the team/organization;
- Software product excellence, including software quality, user experience (UX), and documentation.

Also, both studies showed that AI tools are not in high demand for business and system analysis, but there were many more popular types of tasks for AI tools:

- Coding (including the unit-tests, stored procedures, etc.);
- Code reviews (including the code optimization and refactoring);
- Fast software prototyping.

About 63% of experts in Pashchenko[9] used AI tools in making software code and were satisfied with the reached result.

Both studies demonstrated that the impact of AI tools on software quality management is rising, and the most popular types of tasks for LLMs in software quality management were defined:

- Writing the auto-tests;
- Managing the defects and reports analysis;
- Searching for errors and vulnerabilities in the code.

But still, around half of the experts in Pashchenko[9] do not use AI tools in software quality management.

The expert panel estimated the value and the role of LLMs in learning and in the excellence of software development skills at the end of 2024[8]:

- The impact of using 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 its specific features and risks that we need to estimate before the implementation of AI tools in software production. The expert panel in Pashchenko[9] figured the main barriers to the implementation of any AI-instruments in software development in their teams and organizations:

- High level of different risks: from legal aspects to ethical—44% of experts;
- Lack of 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 Pashchenko[8] to 26% of teams in Pashchenko[9] are executing an official plan on 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[9].

So, Pashchenko[9] confirmed the earlier results from Pashchenko[8]:

1) Innovators' and early adopters' groups are formed in

the software development domain; they have started the AI tools implementation in software production.

2) Current AI-instruments usage is focused on working with the software code (in different kinds of ways), on quality assurance and on the tasks of documentation.
3) For some IT companies, the lack of centralized efforts on the corporate level might lead to the loss of the competitive advantage in software production, connected with AI tools.
4) Software engineers need new skills in AI-human interaction: as earlier companies will start educating them as more effective they will be in the future production process.

It leads to some conclusions about best practices in AI tools implementation:

1. Focus AI usage in software engineering on main process areas in production, starting with all activities where software code and product documentation are the expected result;
2. Formalization of the process of the implementation of AI into software engineering is a key factor in overcoming all barriers;
3. Rising interest in AI from industry and governments leads to the need for 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 Pashchenko [8,9], namely, what difficulties and barriers exist in scenarios for the implementation of AI tools in the activities of a software development company. Of course, the implementation of AI tools has its own characteristics and risks that must be managed on an ongoing basis.

3.2. 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 [8,9], with the objective of developing internal best practices. At the end of 2023,

official policy was established to guide the use of AI tools for solving typical tasks in software production processes, including code, tests and document generation.

Best practices in particular software projects 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 Extensible Markup Language (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[14])—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 [15].

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.

№	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 program code level on the first try. Working versions of the program code—meet 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.	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. Working versions of the program code—meet 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 supplemented them to the working version.	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.
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.

Table 3. *Cont.*

№	Software Engineering Process Area	Current Study Question (from Table 2)	ChatGPT 3.5 Usage Results (2023–2024)	ChatGPT 4 Usage Results (2024–2025)
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 [16]), the team turned to ChatGPT 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 [17]. 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, and 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.

At 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 [18] 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 coordinating the requirements) is an iterative process; it's aligned with other research [19,20]. 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 document generation had been done in a formal structure:
    a. **Role and Context** (for the AI agent in the 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, includ-

ing 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 [21,22], 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 the results of studies by Pashchenko [9] and Naimi et al. [23];
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 et al. [24]. 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 the practical project are aligned with the results of the study by Cao and Huang [25].
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.

4. Conclusions

Applied research [8,9] has identified the implementation features and key aspects of using artificial intelligence tools in the practice of software development companies. Research in 2023–2024 has 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 [8,9] 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.

Within the scope of this study and all limitations de-

scribed above, the scientific problem of identifying the fundamental capabilities of AI tools in producing key software project artifacts across all stages of the software development life cycle was addressed, ranging from technical specifications to executable code and comprehensive user documentation. The use of the AI tool by the development team followed an algorithmic and standardized process defined by corporate policy, encompassing the formulation of precise prompts for the selected ChatGPT tool and the practical application of its validated outputs. Although engineers interacted with the AI tool individually, the resulting artifacts were regularly reviewed, discussed, and refined collaboratively by the entire team.

The ChatGPT tool demonstrated high effectiveness in the development of technical specifications, rapid functional prototyping, and software code generation. Notably, requirements management was maintained throughout the project, enabling flexible adaptation to emerging ideas and continuous, incremental changes in business requirements. The AI tool was employed for engineering consultations, preparation of technical specifications, generation of system code, development of automated tests and test cases, compilation of user documentation, and creation of application content. Based on this experience, the following conclusions were drawn:

- The use of AI for generating technical specifications and user documentation proved to be effective, particularly in the context of service-oriented architectures and in articulating the business value delivered to users by individual services.
- AI-assisted system code generation was successful but inherently iterative, requiring persistent and structured interaction with the AI tool, including repeated reinforcement of context and requirements within prompts. Limitations were observed in the AI tool's ability to consistently retain and reason about the logical interdependencies among system components.
- AI-driven generation of application content was rapid and efficient and did not require additional involvement from external specialists. However, the AI tool exhibited limited capacity for independently developing core business logic and generating original conceptual ideas, necessitating substantial human input.

Overall, the use of ChatGPT met the project team's expectations in most respects, although certain stages required additional attention, intellectual effort, and iterative refinement. Based on the findings, it can be concluded that AI-assisted software development represents an effective approach, provided that it is supported by appropriate skills, ongoing human oversight, and systematic verification of results by software engineers.

Funding

This work received no external funding.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

Not applicable.

Conflicts of Interest

The author declares no conflict of interest.

AI Use Statement

The author declares that no artificial intelligence (AI) tools were used in the preparation of this manuscript.

References

[1] Gorban, A.N., Grechuk, B., Tyukin, I.Y., 2018. Augmented Artificial Intelligence: A Conceptual Framework. arXiv preprint. arXiv:1802.02172. DOI: https://doi.org/10.48550/arXiv.1802.02172

[2] Kästner, C., Kang, E., 2020. Teaching Software Engineering for AI-Enabled Systems. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training, online, 23–29 May 2020; pp. 45–48. DOI: https://doi.org/10.1145/3377814.3381714

[3] Barenkamp, M., Rebstadt, J., Thomas, O., 2020. Applications of AI in Classical Software Engineering. AI Perspectives. 2, 1. DOI: https://doi.org/10.1186/s42467-020-00005-4

[4] Boyko, O., Holoborodko, Y., 2024. Distributed Software Development in 2025: All You Should Know. Available from: https://spd.tech/dedicated-development-teams/distributed-software-development-team/ (cited 24 August 2025).

[5] Pashchenko, D.S., 2024. Consolidation of a New Organizational and Production Paradigm in Software Development Projects. Information Technologies. 3, 150–158. DOI: https://doi.org/10.17587/it.30.150-158

[6] Panetta, K., 2023. Set Up Now for AI to Augment Software Development. Gartner: Stamford, CT, USA.

[7] Cakmak, Z., 2023. Adapting to Environmental Change: The Importance of Organizational Agility in the Business Landscape. Florya Chronicles of Political Economy. 9(1), 67–87. DOI: https://doi.org/10.17932/IAU.FCPE.2015.010/fcpe_v09i1004

[8] 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. DOI: https://doi.org/10.5815/ijitcs.2023.06.03

[9] Pashchenko, D.S., 2025. Growing Demand for Artificial Intelligence Tools in Software Engineering: Results of a Pan-European Study 2024. Revista de Investigación en Tecnologías de la Información. 13(29), 82–91. DOI: https://doi.org/10.36825/RITI.13.29.008 (in Spanish)

[10] Misra, S., Kumar, V., Kumar, U., et al., 2012. Agile Software Development Practices: Evolution, Principles, and Criticisms. International Journal of Quality and Reliability Management. 29, 972–980.

[11] Barai, T.V., Prasad, E., 2023. Smartwatch Market (2023–2032). Available from: https://www.alliedmarketresearch.com/smartwatch-market (cited 24 August 2025).

[12] Ahlgren, M., 2023. OpenAI Statistics and Facts for 2023 (DALL·E, ChatGPT and GPT-3.5). Websiterating: Sunshine Coast, Australia.

[13] Rogers, E.M., 2003. Diffusion of Innovations, 5th ed. Simon & Schuster: New York, NY, USA.

[14] Wierda, G., 2021. Mastering ArchiMate Edition 3.1: A Serious Introduction to the ArchiMate Enterprise Architecture Modeling Language. R&A: Heerlen, The Netherlands.

[15] Sommerville, I., 2009. Software Engineering, 9th ed. Addison-Wesley: Boston, MA, USA.

[16] Chuah, S.H.-W., Rauschnabel, P.A., Krey, N., et al., 2016. Wearable Technologies: The Role of Usefulness and Visibility in Smartwatch Adoption. Computers in Human Behavior. 65, 276–284. DOI: https://doi.org/10.1016/j.chb.2016.07.047

[17] Sattelberg, W., 2021. Google is Updating App Guidelines for Smartwatches Just in Time for Wear OS. Available from: https://www.androidpolice.com/2021/09/03/google-is-updating-app-guidelines-for-smartwatches-just-in-time-for-wear-os-3/ (cited 24 August 2025).

[18] Glinka, F., Raed, A., Gorlatch, S., 2010. A Service-Oriented Interface for Highly Interactive Distributed Applications. In: Lin, H.-X., Alexander, M., Forsell, M., et al. (Eds.). Euro-Par 2009—Parallel Processing Workshops. Springer: Berlin/Heidelberg, Germany. pp. 266–277. DOI: https://doi.org/10.1007/978-3-642-14122-5_31

[19] Ferdiana, R., 2024. The Impact of Artificial Intelligence on Programmer Productivity. In Proceedings of the International Conference on Software Engineering and Information Technology (ICOSEIT) 2024, Bandung, Indonesia, 28–29 February 2024; pp. 1–6. Available from: https://www.researchgate.net/publication/378962192_The_Impact_of_Artificial_Intelligence_on_Programmer_Productivity

[20] Finio, M., Downie, A., 2024. AI in software development. Available from: https://www.ibm.com/es-es/think/topics/ai-in-software-development (cited 24 August 2025). (in Spanish)

[21] Peng, S., Kalliamvakou, E., Cihon, P., et al., 2023. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. Available from: https://arxiv.org/pdf/2302.06590.pdf (cited 24 August 2025).

[22] Petrovic, N., Lebioda, K., Zolfaghari, V., et al., 2024. LLM-Driven Testing for Autonomous Driving Scenarios. In Proceedings of the 2024 2nd International Conference on Foundation and Large Language Models, Dubai, United Arab Emirates, 26–29 November 2024; pp. 173–178.

[23] Naimi, L., Bouziane, E.M., Jakimi, A., et al., 2024. Automating Software Documentation: Employing LLMs for Precise Use Case Description. Procedia Computer Science. 246, 1346–1354. DOI: https://doi.org/10.1016/j.procs.2024.09.568

[24] Kurnianingrum, D., Jumbri, I.A., Ratnapuri, C.I., et al., 2024. Exploring the ChatGPT's Impact and Prospects for Business Research Purposes. In Proceedings of the 2024 3rd International Conference for Innovation in Technology (INOCON), Bangalore, India, 18–20 April 2024; pp. 1–6. DOI: https://doi.org/10.1109/INOCON60754.2024.10511483

[25] Cao, S., Huang, C.-M., 2022. Understanding User Reliance on AI in Assisted Decision-Making. Proceedings of the ACM on Human-Computer Interaction. 6, 471.