Pakistan Journal of Engineering Technology and Science (PJETS)
Volume 12. Issue.02, PP. 91-101, December 2024
https://doi.org/10.22555/pjets.v12I2.1210
E-ISSN 2224-2333
P-ISSN 2222-9930

Code and Beyond: A Review on the Impact of AI on Modern Software Development

Aneeta Lohana [1], Sumbul Ghulamani [1*], Kamran khowaja[1], Kazim Raza Talpur[2], Asadullah Shah[3]
[1]Department of Computer Sceicne, SZABIST University, Hyderabad Campus, Pakistan
[2]Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia, Malaysia
[3]Kulliyah of Information and Communication Technology, International Islamic University Malaysia, Malaysia

[*]Corresponding Author: sumbul.ghulamani@hyd.szabist.edu.pk

Abstract:

This comprehensive review examines the link between AI and software development by examining their interplay throughout all development life cycle stages. The review incorporates information from recent research and industry practices to explain how AI affects processes like requirement analysis, design, coding, and testing. The research uses the Review Paper technique for examination, covering a wide range of sources, including academic databases, research portals, and publications. The seven-week timeframe includes data analysis, a study of the literature, and a critical examination of the implications and effective role of AI. Furthermore, the findings indicate that AI iactus is a collaborative tool in software development, based on historical and contemporary technical realities. The analysis not only gives more detailed knowledge but also establishes the framework for future research. Underlining the need for industry-specific AI applications and a better evaluation of developers' experiences. This research acts as a guidepost for industry executives and academics by providing strategic recommendations for navigating the evolving role of AI in software development.

Keywords: *Artificial intelligence, Automation in programming, Code optimization, Computer intelligence, Knowledge-based systems, Software engineering, Software development*

I. INTRODUCTION

Software development has undergone significant transformations in recent years, driven by the widespread integration of Artificial Intelligence (AI) technologies. AI's fast development has altered how developers deal with challenges in the software development life cycle. By delving into how these technologies are transforming the software lifecycle, it becomes crucial to understand the various practices and techniques that utilize AI to address both traditional and emerging issues in this field. The field of software development is the baseline of a technological shift in almost every sector, particularly during the COVID-19 era when there was an unparalleled shift to remote work. Although many sectors were unfamiliar with the concept of remote work, software professionals have long been used to it also during the pandemic, there has been a considerable increase in the demand for software developers [1] This rise is not just a result of a temporary trend, but rather demonstrates the importance of software development across a variety of sectors such as healthcare, education, banking, and other industries. These sectors depend on developers to customize software solutions and overcome the pandemic's obstacles. Software developers' adaptability and reactivity have been proven effective in providing creative solutions adapted to the specific needs of each sector [2].

However, Incorporating intelligent systems into the development lifecycle raises interesting questions about the future of the field. Software creation has always been at the forefront of innovation, creating solutions that empower other industries. As the digital era progresses, the landscape of technology undergoes a profound transformation, from then the most emerging field is AI and its subsets. These intelligent entities can understand external data and do activities with the semblance of intellect. Additionally, they are set to change the technological landscape. The market dynamics reflect this trend, with the worldwide AI software industry expected to rise rapidly. According to the significant market research organization Tractica, sales in this arena are expected to increase from around 9.5 billion US dollars in 2018

to a stunning 118.6 billion US dollars by 2025 [3]. Apple, Google, and Amazon, among others, prominently integrate AI into their product launches while also investing in strategic acquisitions of AI-based firms. In 2019, the adoption of AI has tripled in 12 months, which could make it the fastest paradigm shift in technology history [3].

The effect of automation tools on the development workflow involves a continuous exchange between positive advances and nuanced challenges. In line with the realm of positive influence, AI technologies like ChatGPT, GitHub Co-pilot, and Codex emerge as invaluable tools, streamlining various features of software development. Such as when it comes to code merging which was a very difficult procedure in the past is now sped up with GitHub Co-pilot, increasing productivity and decreasing human labor. Contrarily, Codex transforms code creation by providing creative answers to challenging programming problems [4]. These applications serve as excellent examples of how AI may help developers save time. However, as with any transformative force, the impact of AI on software development introduces specific challenges. These include ethical considerations, potential biases in AI algorithms, and the need to maintain a delicate balance between human creativity and machine efficiency this prompts us to navigate these challenges thoroughly, ensuring that the positive impact is maximized while mitigating any adverse effects. Recent research, [4] further sheds light on the limitations of AI, particularly in addressing the challenge of fixing large code bugs. The research focused on the Codex AI tool and concluded that, despite Codex's impressive capabilities in synthesizing code snippets, it faces challenges in effectively localizing and resolving problems. The findings reveal that, while Codex demonstrates surprising effectiveness, especially in Python, it falls short of providing a comprehensive bug-fixing solution. This underscores the crucial significance of human developers in complex problem-solving circumstances.

The goal of this research is to gain insight into all of the variables underlying the link of transformation between AI and Software development. This can be done by thoroughly investigating how the integration of AI technologies affects the traditional paradigms of software development processes and by closely examining important techniques and technologies, Research also focuses on determining the dual influence of AI as a collaborative tool. Ultimately, this research aims to clarify how AI affects, enhances, or even completely rewrites accepted methods in the software development lifecycle.

II. LITERATURE REVIEW

This section explores the dynamic interplay between AI and software development, to begin with the exploration, it is necessary to go into the technical history where paradigm-shifting inventions often become the point of researching job replacements and industry transformations. The invention of cloud computing marked one such example; according to research, there is a shifting trend in employment needs, as seen by ads for technical jobs. Ads for system administrator and technical support roles increased sharply from 1988 to 1995, then declined from 1995 to 2001 [5]. This tendency corresponds to the advent of networked computers in businesses. As a result, there has been a noticeable trend toward advertising for network administrators and, more recently, cloud architects, emphasizing the dynamic growth of positions. The field made a wonderful impact on Information Technology (IT) over the past few years, huge companies such as Google, Amazon, and Microsoft struggle to provide more powerful, dependable, and cost-efficient cloud platforms, and business organizations seek to reshape their business models to gain benefit from this new paradigm. Studies indicate that nearly 14 million new jobs are expected to emerge between 2011 and 2020 due to the influence of cloud technology [6]. This indicates that the introduction of technology doesn’t replace the roles but it just transforms the roles effectively, and the person involved in a particular field needed to update the skill set and adapt to the new technological environment.

Another example, which shows the paradigm shift is the program generators. The introduction of program generators was an important point in software development because it brought the new concept of automation and efficiency. However, its acceptance was met with some criticism in the programming community at first mentioned in the research [7]. This Research states that some programmers were fearful of this attempt at automating their job; in fact, many became resentful because of some unfortunate reasons at that time but as they began using the program generator, this fear gradually faded as they realized that, even though they are automated, they still need human input. In situations where they need to solve problems, they also noticed that the programmer usually spends more time thinking about how to communicate a problem to the computer than actually solving it [7]. Furthermore, in the advancements of the

roadmap program generators, the low code or no code platforms developed which [8] have enabled people with varied technical backgrounds to actively participate in software development. These platforms make application development more accessible by allowing business analysts, domain experts, and citizen developers to engage in the development process. Rather than replacing the necessity for professional software developers, these platforms have broadened the range of people who can contribute to the development of software solutions. Professional developers have evolved by focusing on more difficult and detailed areas of software design, issue resolution, and optimization, while non-developers help with speedy application development. Hence program generators and low code platforms path illustrates the interplay between technology and human innovation. While there were doubts at the beginning, they gradually changed and things moved towards acceptance and collaboration, emphasizing the critical importance of human knowledge in the ever-changing world of software development.

Moreover, when it comes to the technologies that offer automation and efficiency the one widely used today is ATMs. The introduction of ATMs and other back-office processing technology in the banking business had a dramatic effect. Through improved software solutions, these technologies decreased the costs involved with individual transactions while also improving the administration of huge amounts of client data. Simultaneously, new channels for the delivery of retail financial services evolved, such as telephone banking and PC-based banking [9]. Further, it is also mentioned in the research [9] that these employing "transaction-based approaches," aimed to enhance efficiency by replacing traditional human tellers' services with ATMs and telephone banking and that it is the point where many bank tellers' jobs obsolete, owing to fears over employment losses. However, the truth was rather different. ATMs, rather than displacing human tellers, have resulted in a redefining of their duties. Routine transactions were automated, allowing tellers to focus on more complicated client needs including financial advice, account management, and personalized services. The banking industry's adaption led to the emergence of new professional categories, such as customer relationship managers, financial advisers, and digital banking technology professionals. This historical example demonstrates that technology developments, rather than destroying occupations, frequently result in role rethinking and the introduction of new opportunities within a sector.

However, a consistent pattern emerges from the study of diverse technical breakthroughs such as cloud computing, program generators, and ATMs. While these developments originally raised worries about job displacement, the historical record shows with research [9] [7], [10] [5] that technology has been more effective at modifying employment rather than completely replacing it. Moreover, AI is also being used by high-tech firms like Facebook and Ubisoft to increase software stability and expedite issue detection. Facebook's artificial intelligence (AI) technologies examine vast amounts of code, spotting possible errors and offering solutions. Technologies such as Sapienz use automated testing to find and fix problems in mobile apps quickly. Similar to this, Ubisoft uses AI to examine code and gameplay data to spot irregularities in its intricate games that conventional debugging might overlook. These AI-powered methods not only expedite development schedules and enhance software quality but also free up developers to concentrate on strategic and creative work, eventually resulting in more inventive and reliable products[11].

Hence, the development of roles, the introduction of new job opportunities, and the ongoing need for human oversight in specialized areas demonstrate the workforce's adaptability and flexibility in technological progress. Rather than closing doors, technology has continuously opened new ones, allowing for skill development, specialization, and advancement in parallel with the changing landscape of the industry.

This emphasizes the symbiotic link between technology and human potential, emphasizing opportunity enhancement rather than reduction. In line with the same pattern, the emergence of AI, in the field of software development will increase the opportunities rather than replace the field.

A. The Interplay of AI in the Software Development Process

As far as historical backgrounds are concerned, it is demonstrated that whenever a new technology is introduced, it invariably creates room for new jobs and opportunities, this analogue also applies to Artificial Intelligence (AI) and software development. Some challenges associated with developed software and its related data, such as source code, modifications, bug reports, and user feedback, can be costly. These include conceptual specifying, designing, and testing the software, as well as representing reliability and telemetry information, all of which must be created from scratch. Therefore, selecting appropriate programming technology is crucial as it can enhance development efficiency

and ultimately reduce software costs [12]. In the software development field, there's a risk of code crashing unexpectedly. For instance, a program may work perfectly fine, but suddenly stop functioning after making no changes. This can pose challenges in tracing and identifying the underlying issue. So, to keep this necessity in account and cope with these challenges AI is offering many solutions through its subsets, those subsets include neural networks, fuzzy logic, knowledge-based systems, and many more. These subsets are specifically merged in the development process, and each stage of the development process is now evolved through AI. By, including AI in the process increases overall software development process reusability and promotes automation for company standards. Following is the section that describes how AI subsets are aimed to increase the software development process at each step and Table 1 is the summarization of all subsets that are used in each of the software development phases. The following section describes how AI subsets are aimed to increase the software development process at each step.

Planning: This phase includes a Feasibility Study, Cost Estimation Risk Assessment, Scheduling, Resource Planning, Quality Assurance Planning, Communication Planning, Management, and Documentation Planning[13]. AI's subset Knowledge-based systems (KBS) aim to capitalize on project planning expertise and improve future planning by storing and using previous experiences [14]. Some studies explored some representation schemes such as SRL (Structured Representation Language) for project planning ideas [15], on the other hand, neural networks (NNs) have proven effective in forecasting outcomes, particularly in risk assessment [16]. Previously Backpropagation and genetic algorithms were used to train NNs in studies that indicated their usefulness in forecasting hazards connected with software development projects[16]. Furthermore, in project scheduling, genetic algorithms (GAs) have been widely used, converting planning into requirement satisfaction issues with optimization targets[17]. They are mainly used to evolve methods for estimating software effort and producing optimal schedules and job allocations. These are also very useful to improve schedules by considering trade-offs between aspects such as cost, length, and quality[17]. Additionally, Case-based reasoning (CBR) is used to explore the integration of previous project experiences into project planning [14].

Requirement Analysis: The requirement analysis phase in software development involves several subtasks aimed at understanding, documenting, and validating the needs and constraints of the stakeholders.[13] But, to develop the strong roots that allow the software to grow, the second phase of exploration requires a deeper understanding of problem and solution characteristics [18] and this is the main reason that in this phase both stakeholders and team meet together and analyze the facts and figures. However, due to the involvement of stakeholders in this phase, several ambiguities arise. These include communication problems, misunderstandings during requirement gathering, and uncertainty about the requirements themselves. Such issues are costly as this phase sets the baseline for development. To address these challenges, AI techniques prove effective. For example, Knowledge-based systems (KBS), such as READS technology, play a crucial role in guiding the complex process of requirement identification, analysis, and decomposition [14] [19]. These systems begin by displaying requirements documents, allowing for both manual and automatic identification. The found needs are carefully edited, reviewed, allocated, and reduced to provide a collection of unambiguous assertions. Furthermore, in the case of conceptualization requirements, ontologies offer a systematic framework for collecting and organizing knowledge, facilitating interoperability, and improving semantic representation [14]. Additionally, there is no direct application of concepts such as continuity, which is highly obvious and useful in the physical world. Small modifications to requirements in software engineering can result in far-reaching and radical changes in the overall cost of the project [20]. So computational Intelligence (CI) is highly useful also it helps in using case-based reasoning. The approaches, such as the SPECIFIER system are used to solve the limitation [14]. They enable us to efficiently traverse competing needs and assess priorities. Hence the incorporation of AI in requirements engineering not only speeds up the difficult process but also helps to minimize mistakes and maximize software development efficiency

Design And Development: This phase includes System Design, Prototyping, Debugging, Documentation, Version Control, and Deployment this is the practical phase [13]. In this phase, it is needed to decide how much programming work is required to build a user interface, and how frequently it has to be altered [9]. The user interface must represent the underlying data semantics and respond appropriately to user inputs these are included in the challenges of this phase. So, as per AI is concerned to solve these problems technologies such as code generation are used which helps to generate the code, fast and effectively, The one AI-powered tool which is Codex is helpful in the Deployment of

Code. The Codex is trained on more than 50 million GitHub repositories including the vast majority of GitHub Python code, totaling 159 GB [21]. The codex takes the English language as the prompts input and generates the code in several languages such as GO, PERAL, PHP, RUBY, and SWIFT, but it works best on Python. Another AI-powered tool that helps in the process of version control, is the GitHub Copilot this tool has a tagline "your AI pair programmer" [21]. This tool is the cooperation between GitHub and Open AI, and it has transformed the coding and software development practices. This AI-powered code completion tool goes beyond typical autocompletion by predicting entire lines or blocks of code based on context and user input, saving time while also reducing the likelihood of syntax mistakes [21]. Context-aware recommendations, support for different programming languages, ongoing learning from usage, and seamless connection with major Integrated Development Environments (IDEs) like Visual Studio Code are among its primary features. This tool helps modern software development because of its capacity to comprehend context, learn from developer interactions, and improve cooperation by offering context-aligned code snippets. Copilot provides developers with greater coding speed, code exploration for learning, and improved collaboration [21].

Testing and Evaluation: This phase validates the other phases including planning, requirement analysis, design, and development. The testing and evaluation phase comprises various subphases, including the unit test, which validates small blocks of code. Additionally, there is a "monkey test," which evaluates fuzzy logic and inappropriate methods [22]. Furthermore, the testing phase requires the additional cost of hiring the tester and then checking and validating the design and deployment, but with the help of AI technology, this cost is reduced through the different approaches of Knowledge-based systems such as Prolog-based expert system, and REQSPERT, which aims to support test plan development from requirements by classifying them and proposing suitable metrics and tools [14]. Genetic Algorithms (GAs) are recognized as an emerging area of interest in testing optimization [23]. Several studies employ GAs to generate ideal test cases, taking into account issues such as user interface testing, mutation testing, and testing object-oriented applications. The usefulness of GAs is examined, as well as obstacles including insecurity and the need for flexibility [14].

Summary: Hence, an in-depth analysis of the various stages of the software development lifecycle demonstrates the vital function that Artificial Intelligence (AI) plays as a collaborative tool. Instead of displacing developers, AI technologies have emerged as effective companions, increasing productivity and speeding up the whole development process. AI solutions contribute at every step, from planning and requirement analysis to design, development, testing, and evaluation, eliminating obstacles and allowing developers to focus on key issues. This collaborative interplay is especially visible in activities like code optimization, design element recommendations, and promoting effortless interaction with stakeholders. AI not only improves operations but also creates opportunities for creative problem-solving, demonstrating its ability to change the process of software development without replacing the important human touch. The adoption of these technological improvements will enhance the symbiotic link between software development and AI. This collaboration between the two technologies enables a balanced combination of creativity, speed, and precision.

III. DISCUSSION

The interaction between AI and software development provides a symbiotic link that improves the efficiency of software solutions and speeds up problem-solving. As shown in Figure 1 the common words and ideas related to software development and AI working together. It illustrates how AI helps developers solve hard problems and make things work better without taking over their jobs. Instead of replacing humans, AI works alongside them to improve their skills in software development. This figure points out that AI and human intelligence work together to make things better. [24].Furthermore, the outcomes of AI approaches are frequently evaluated or validated by humans. Such user input might be used to develop AI algorithms, creating a continual feedback loop of human and artificial intelligence [24]. Also in the recent two decades, there has been a huge increase in the number of projects and publications involving the use of artificial intelligence approaches to software development [25]. There are several conferences and journals dedicated to publishing research on this topic. Because AI approaches are offered to shorten the time to market and improve the quality of software systems [26]. While the advent of new technology frequently raises worries about future replacement, history has shown that such fears are unjustified [6], [7], [9], [10]. These technologies, on the

other hand, welcome new challenges and possibilities, encouraging developers to improve and update their skill sets. It is argued in the research [27] that the fusion of these two disciplines will be needed for many new software demands.

Table 1: Summarization of the Role of AI in Software Development

Software Development Phase	AI Subsets Used	Applications
Planning	**Knowledge-based Systems, Neural networks, Genetic algorithms, Case-based reasoning**	**- Knowledge-based Systems improve project planning by using past experiences. - Neural networks assist in forecasting outcomes, especially for risk assessment. - Genetic algorithms optimize planning through requirement satisfaction and scheduling. - Case-based reasoning integrates past project experiences into planning.**
Requirement Analysis	**Knowledge-based systems, Ontologies, Computational Intelligence, Case-based reasoning**	**- Knowledge-based systems like READS technology guide requirement identification and analysis. - Ontologies provide a systematic framework for knowledge collection and representation. - Computational Intelligence aids in navigating competing needs and assessing priorities.**
Design and Development	**AI-powered code generation tools**	**- Codex generates code from English prompts, working best with Python. - GitHub Copilot, a code completion tool, suggests context-aware code snippets to enhance collaboration and coding speed.**
Testing and Evaluation	**Knowledge-based systems, Genetic algorithms**	**- Knowledge-based systems support test plan development by classifying requirements and suggesting metrics and tools. - Genetic algorithms optimize testing by generating ideal test cases and addressing issues like user interface testing and mutation testing.**

So, Improving the skillset is very important as mentioned in the research [28] and advances in artificial intelligence (AI) will impact the workplace and role of the developers. The author of the study [28] argues that the importance of AI lies in the feature of making numerous and uncostly predictions. This is because AI technology, particularly machine learning, has decreased the cost of prediction-enhancing algorithms, computing speed, data storage, and retrieval. As a result, incredible progress has been made in image identification and language translation. As prediction jobs grow more automated, judgment abilities will become increasingly useful. Developers must grasp how to successfully utilize AI, acknowledging the complementary nature of prediction and judgment in decision-making.
Furthermore, there is an expression “no update is the big update" that resonates in the ever-changing world of software development, underlining the significance of continual learning and adaptation to harness the full potential of AI in creating the future of software development [29]. The continual learning and adaption hold more importance and are experimentally illustrated from a comprehensive research study [29] which surveyed nine questions to determine the impact of AI on software development. The study circulates the survey via LinkedIn profiles, and responders came from well-known firms such as Apple, Google, Microsoft, Facebook, IBM, Amazon, Adobe, Dell, Boeing, Oracle,

Samsung, and Intel. This varied representation of developers from these repudiated industries highlights the widespread use of AI technologies in their workplace. Crucially, when queried about the possibility of software maintenance, debugging, code fixing without human intervention, and the idea of anyone coding without specialization. The majority of the answer was negative (no). This indicates that, while AI improves efficiency, it does not replace the indispensable human touch in software development.

Moreover, respondents were asked whether writing from scratch will become redundant, with a move toward extensive reuse of existing code and concepts. The majority of respondents said yes, confirming that AI's efficiency resides in reducing the time-consuming aspect of code beginning from scratch. This is consistent with the current trend in coding methods, in which the reuse of existing, reusable code is regarded as critical. Notably, this reflects the fundamental notion addressed by the Java programming language, which sought to minimize the difficulties associated with starting from scratch. Moreover, another research study [29] explains a more in-depth analysis of the future environment of software development and AI and the study concludes the same point that no one can replace the developers unless they want to be so, from the predictions of research [29] it is also highlighted that developers will continue to have a role in the future, either as moderators or creators of AI systems, but only if they can keep themselves upgraded with the latest advancements. Furthermore, in the research study [30], it is stated that Machine Learning (ML) systems have widely been adopted by companies in all industries worldwide as value propositions to create or extend the services and products they offer. In addition, another study highlights the necessity of using machine learning (ML) which is a subset of AI to cope with challenges in software development [31]. The author Elizamary Nascimento [31]surveyed Microsoft teams to understand software development processes and identify essential development challenges. They found that data management and availability, customization and reuse of ML models, and handling ML model interactions are key areas of concern and it is very beneficial in dealing with software development. The author [31] also highlights the importance of education and training on AI and ML techniques for software teams. And mainly research also sheds light that customization and reuse of ML models require different skills than traditional software development and for that, there is a high need for software developers to upgrade themselves with the touch of AI because it is high. However, the path of increasing efficiency is also upgrading such as research studies also put light on the tool parse web which is an AI tool. PARSE Web is a web-based application designed to let programmers reuse existing frameworks or modules. It accepts "Source Destination" searches and offers method-invocation sequences that can change the Source object type into the Destination object type. The program uses code search engines to collect relevant code samples and then does static analysis to extract needed sequences. It also ranks and clusters the recommended sequences using various criteria. PARSE Web evaluations suggest that it is successful in answering programmers' inquiries and outperforms other tools such as Prospector and Strathcona. PARSE Web is a helpful resource for programmers who want to save time and increase efficiency throughout the software development process. Additionally, following the same path of significance of AI, the article [32] also sheds light on AI by stating that AI enhances the knowledge rather than replacing professional expertise its, usage will increase and polish the skills rather than replacing it.

The Latest advancements are not only necessary for the developers who are already involved in this field but it is also necessary for the upcoming developers because in the field of information technology students are asked to solve problems through software systems that they did not design or rebuild, the existing systems. These practices are common in the academia of IT as they increase organizational knowledge and practices and that’s why it is recommended to train the upcoming developers in a way so that they have better knowledge about both the field of AI and the development field [33]. Various AI technologies are expanding very fast such as ChatGPT, Codex, and GithubCopoilt. But from them, ChatGPT has received substantial attention and is frequently used among these technologies, particularly in educational institutions where students seek assistance with their homework. ChatGPT, an advanced language model developed by Open AI, is a unique AI technology with significant potential in the education industry stated in research [34]. Also, this research study draws attention to important findings from a recent survey [34], which found that 91% of company leaders participating in hiring procedures actively seek applicants with ChatGPT expertise in departments such as software engineering, customer support, human resources, and marketing. Furthermore, another research study [35] is along the same line and emphasizes more about the extensive use application of AI chatbots,

such as ChatGPT and Google Bard, across various educational institutions. These applications have one of its distinguishing advantages which is the capacity to provide personalized instructions, and that feature solves a variety of difficulties in the educational landscape. Also, the study [36] offers insight into the dual nature of possibilities and problems that develop in scenarios such as software engineers being evaluated through coding challenges or recruits integrating into an organization's business and technical processes. The possibility of using ChatGPT to help developers create more robust software in less time is seen as a possible gain. However, the study underlines the important point that educators are responsible for teaching the required skills to software engineers and that involves not just properly utilizing technology but also offering the capacity to manage and evaluate these tools [36]. It also emphasizes the necessity of developing a clear and well-defined track of activities in the educational process, ensuring that technology adoption matches with the larger aims of education. So, both the studies [34] and [36] have the common point that it is necessary to train the upcoming developers with the whole basis of development and also with the skills for the use of the AI tools in their core.

Besides the advantages of AI, several ethical concerns need to be addressed. One significant issue is job displacement, particularly for low-skilled workers, as AI and automation can replace routine tasks. While AI offers opportunities for new employment and productivity, it also creates challenges for the workforce[37]. Additionally, privacy concerns arise, particularly with AI devices that have built-in cameras. Research has shown that humanoid AI devices tend to trigger stronger privacy concerns, especially in private settings and among women[38]. As AI continues to evolve, it is crucial to consider these ethical issues and implement responsible practices to ensure the technology benefits society as a whole.

Figure 1: Word cloud of AI and Software development

A. Limitations of AI in the field of development

Excessive dependence on pre-existing patterns: AI development tools may rely too much on known patterns in training data, thereby restricting the invention of new solutions that might be more effective for a problem

Bias in Coding: AI algorithms used for code creation may unintentionally incorporate biases present in the training data, resulting in the reinforcement of specific coding styles and methods and potentially limiting diversity in programming techniques. This limitation is also highlighted in the research [11] where Codex was able to solve the bug up to 50% in the language Python.

Difficulty in Dealing with Ambiguity: In software development, AI systems may struggle to manage ambiguous requirements or specifications, necessitating human intervention to offer clarity and context.

Problem-Solving with Limited Creativity: While AI technologies may automate typical coding chores, they may lack the creative problem-solving talents that human coders bring to the table, and that is also the case when testing codex, it limits the problem-solving ability and hence cannot able to debug large code [38].

The Difficulty of Making Complex Decisions: AI may confront difficulties in making complicated judgments throughout the software development process when a thorough grasp of project goals, user needs, and the larger business environment is essential [39].

Unstructured Data Interpretation: Unstructured data, such as free-form user comments or open-ended project specs, may be difficult for AI technologies to analyze, making it difficult to extract significant insights from qualitative data. Additionally, the research [40] experimentally proved that since codex is dependent only on GitHub code there is the limitation of unstructured data which as a result cannot solve the complex problem.

Handling Unexpected Scenarios Inadequately: In software development, AI models may not be able to handle unexpected or outlier events successfully, forcing developers to intervene and overcome unforeseen issues.

Problems with Integration and Compatibility: Integrating AI technologies into existing development workflows might cause integration difficulties and compatibility concerns with other tools, frameworks, or systems that developers use.

Adoption Difficulties and the Learning Curve: Adopting new AI technologies may require a learning curve for developers, and the first phases of integration may disrupt established workflows before the advantages become evident.

Dependence on Training Data Quality: The quality and representativeness of the training data are closely related to the efficacy of AI technologies in code development. In circumstances when the data is biased or lacks diversity, the resultant code may not adhere to recommended standards [40].

Implications for Ethics and Law: Developers who employ AI technologies must manage ethical considerations such as privacy, security, and responsible AI use. Adherence to legal frameworks and ethical development methods becomes crucial [41].

Continuous Supervision is Required: To guarantee that the resulting code matches project goals and quality standards, AI technologies may need continual supervision and validation by developers, particularly in important decision-making processes.

IV. CONCLUSION

In conclusion, this study utilizes an effective Review Paper technique to give a thorough examination of the interaction between Artificial Intelligence (AI) and software development. The research explores AI's impact on software development by reviewing literature and industry practices. Moreover, it analyzes AI technologies in software development, highlighting their benefits like increased productivity, efficiency, and new development approaches. Concurrently, it thoroughly assessed the challenges associated with AI integration, such as ethical constraints, unstructured data interpretation, possible biases, and many more. The study also intended to draw important insights that will help industry executives and academics navigate the emerging convergence of AI and software development through this detailed examination.

Moreover, the central theme of the research highlights the collaborative function of AI in combination with human developers as a key topic. Rather than replacing humans, the analysis highlights how AI, through tools like ChatGPT, Codex, and GitHub Copilot, enhances productivity and allows developers to focus on more creative aspects of their work The interplay between AI and human developers becomes clear, with AI relieving operational constraints while developers leverage their expertise for complex problem-solving. After that, the study dispels common misconceptions about AI. It dispels the myth that AI can predict the future, explaining its limitations due to past data, developer assumptions, and the unpredictability of events. The findings underline the importance of human judgment in interpreting AI forecasts and highlight the need for continuous learning and skill development within the developer community. As AI evolves, developers must update their skills. Tools like ChatGPT, Codex, and GitHub Copilot should be integrated into education for hands-on experience. The study highlights the growing demand for ChatGPT skills across industries and recommends including AI technologies in education to prepare the next generation for an AI-driven future. However, acknowledging AI's significant progress in creative activities, the study clarifies the idea that AI lacks creativity. The challenges traditional views of creativity by showing how AI creates unique outputs in music, poetry, and visual arts, positioning AI as a driver of expanded creativity. However, it has also been witnessed in the recent two decades that there is an enormous increase in software product complexity and size, as well as an unexpected growth of application areas. The software has gotten increasingly "intelligent", "connected", and "mobile". Artificial Intelligence (AI) developments have made their way into current applications to tackle more complex issues, deliver better solutions (optimized), or learn from their behavior. These factors highlight the increasing need for software developers to design AI-based applications that solve real-world and societal problems, demonstrating the strong rela-

tionship between AI and software development, and offering more opportunities for future research. Hence these insights answer the research question by, highlighting AI, as a collaborative tool rather than replacing developers' positions, instead, it enhances their capabilities and assists them in overcoming challenges.

Finally, as the world approaches an AI-powered future, this study acts as a guidepost, giving nuanced insights for industry executives, educators, and developers alike. Responsible AI incorporation into software development demands an awareness of its capabilities, ethical issues, and continual improvements. This research helps to build a collaborative, imaginative, and ethical future in the ever-changing field of AI-infused software development.

REFERENCES

[1] T. Breaux and J. Moritz, "The 2021 software developer shortage is coming: Companies must address the difficulty of hiring and retaining high-skilled employees from an increasingly smaller labor supply," Jun. 21, 2021, Association for Computing Machinery. doi: 10.1145/3440753.

[2] A. Günsel, A. Açikgšz, A. Tükel, and E. Öğüt, "The Role Of Flexibility On Software Development Performance: An Empirical Study On Software Development Teams," Procedia Soc Behav Sci, vol. 58, pp. 853–860, Oct. 2012, doi: 10.1016/j.sbspro.2012.09.1063.

[3] A. Nguyen-Duc, I. Sundbø, E. Nascimento, T. Conte, I. Ahmed, and P. Abrahamsson, "A Multiple Case Study of Artificial Intelligent System Development in Industry," in ACM International Conference Proceeding Series, Association for Computing Machinery, Apr. 2020, pp. 1–10. doi: 10.1145/3383219.3383220.

[4] Julian Aron Prenner, Hlib Babii, and Romain Robbes, "Can OpenAI's Codex Fix Bugs?: An evaluation on QuixBugs," IEEE, May 2022.

[5] M. J. Gallivan, D. P. Truex III, and L. Kvasny, "Changing Patterns in IT Skill Sets 1988-2003: A Content Analysis of Classified Advertising," 2002.

[6] U. S. Singh and K. Gulati, "Our Heritage CLOUD COMPUTING: A REVOLUTION IN JOB CREATION AND JOB DESIGN," 2019. [Online]. Available: https://www.researchgate.net/publication/351990886

[7] R. L. Roth, "Program generators and their effect on programmer productivity."

[8] K.-D. Althoff and J. Rech, "Artificial Intelligence and Software Engineering: Status and Future Trends," 2004. [Online]. Available: https://www.researchgate.net/publication/220633840

[9] L. Hunter, A. Bernhardt, K. L. Hughes, and E. Skuratowicz, "IT'S NOT JUST THE ATMs: TECHNOLOGY, FIRM STRATEGIES, JOBS, AND EARNINGS IN RETAIL BANKING." [Online]. Available: www.jstor.org

[10] T. Al-Rousan, "Cloud computing for global software development: Opportunities and challenges," in Transportation Systems and Engineering: Concepts, Methodologies, Tools, and Applications, vol. 2–3, IGI Global, 2015, pp. 897–908. doi: 10.4018/978-1-4666-8473-7.ch045.

[11] Daniel Ajiga, Patrick Azuka Okeleke, Samuel Olaoluwa Folorunsho, and Chinedu Ezeigweneme, "Enhancing software development practices with AI insights in high-tech companies," Computer Science & IT Research Journal, vol. 5, no. 8, pp. 1897–1919, Aug. 2024, doi: 10.51594/csitrj.v5i8.1450.

[12] M. Kim, T. Zimmermann, R. DeLine, and A. Begel, "The emerging role of data scientists on software development teams," in Proceedings - International Conference on Software Engineering, IEEE Computer Society, May 2016, pp. 96–107. doi: 10.1145/2884781.2884783.

[13] P. Venkateswara, R. V Pradeep Kumar, B. Pradeep, and K. Reddy, "Applying Agile Software Methodology for the Development of Software Development Life Cycle Process (SDLC)," Journal for Research, vol. 04, 2018, [Online]. Available: www.journal4research.org

[14] F. Meziane and S. Vadera, "Artificial Intelligence in Software Engineering," 2010, pp. 278–299. doi: 10.4018/978-1-60566-758-4.ch014.

[15] Y. Gil and J. Blythe, "How Can a Structured Representation of Capabilities Help in Planning?," 2000. [Online]. Available: www.aaai.org

[16] W. Hua, W. Ben, Y. Longyong, and Z. Wentong, "Software Risk Assessment Method based on Fuzzy Neural Network," 2015.

[17] L. Ozdamar, "A Genetic Algorithm Approach to a General Category Project Scheduling Problem," 1999. [Online]. Available: https://www.researchgate.net/publication/220509584

[18] M. Harman, "The Current State and Future of Search Based Software Engineering," 2007.

[19] Wemembu UR, Okonta OE, Imiere EE, Ajani D, and Owolabi AA, "Knowledge-Based Management System and Dearth of Flexible Framework for Software Development," 2015.

[20] W. Pedrycz, "Computational Intelligence as an Emerging Paradigm of Software Engineering," 2002.

[21] B. A. Becker, P. Denny, J. Finnie-Ansley, A. Luxton-Reilly, J. Prather, and E. A. Santos, "Programming Is Hard - or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation," in SIGCSE 2023 - Proceedings of the 54th ACM Technical Symposium on Computer Science Education, Association for Computing Machinery, Inc, Mar. 2023, pp. 500–506. doi: 10.1145/3545945.3569759.

[22] L. Luo, "Software Testing Techniques Technology Maturation and Research Strategy Class Report for 17-939A Software Testing Techniques Technology Maturation and Research Strategies."

[23] A. Sharma, R. Patani, and A. Aggarwal, "Software Testing Using Genetic Algorithms," International Journal of Computer Science & Engineering Survey, vol. 7, no. 2, pp. 21–33, Apr. 2016, doi: 10.5121/ijcses.2016.7203.

[24] 2013 2nd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE) : proceedings : May 25-26, 2013, San Francisco, CA, USA. IEEE, 2013.

[25] S. Martínez-Fernández et al., "Software Engineering for AI-Based Systems: A Survey," May 2021, doi: 10.1145/3487043.

[26] H. Ammar, M. S. Hamdi, H. H. Ammar, and W. Abdelmoez, "Software Engineering Using Artificial Intelligence Techniques: Current State and Open Problems," 2012. [Online]. Available: https://www.researchgate.net/publication/254198356

[27] L. Ford, "Artificial intelligence and software engineering: a tutorial introduction to their relationship," 1987.

[28] A. Agrawal, J. S. Gans, and A. Goldfarb, "What to Expect From Artificial Intelligence." [Online]. Available: http://mitsmr.com/2jZdf1Y

[29] L. Mahmoud and A. Zohair, "The Future of Software Engineering by 2050s: Will AI Replace Software Engineers?," 2018. [Online]. Available: http://journals.sfu.ca/ijitls

[30] O. A. Bastías, J. Díaz, and J. López Fenner, "Exploring the Intersection between Software Maintenance and Machine Learning—A Systematic Mapping Study," Applied Sciences (Switzerland), vol. 13, no. 3, Feb. 2023, doi: 10.3390/app13031710.

[31] E. Nascimento, A. Nguyen-Duc, I. Sundbø, and T. Conte, "Software engineering for artificial intelligence and machine learning software: A systematic literature review."

[32] M. Hatfield, "Professionally Responsible Artificial Intelligence," 2019. [Online]. Available: https://digitalcommons.law.uw.edu/faculty-articleshttps://digitalcommons.law.uw.edu/faculty-articles/540

[33] D. Kalles, "Artificial intelligence meets software engineering in computing education," in ACM International Conference Proceeding Series, Association for Computing Machinery, May 2016. doi: 10.1145/2903220.2903223.

[34] O. Tayan, A. Hassan, K. Khankan, and S. Askool, "Considerations for adapting higher education technology courses for AI large language models: A critical review of the impact of ChatGPT," Machine Learning with Applications, vol. 15, p. 100513, Mar. 2024, doi: 10.1016/j.mlwa.2023.100513.

[35] L. Diosan and S. Motogna, "Artificial intelligence meets software engineering in the classroom," in EASEAI 2019 - Proceedings of the 1st ACM SIGSOFT International Workshop on Education through Advanced Software Engineering and Artificial Intelligence, co-located with ESEC/FSE 2019, Association for Computing Machinery, Inc, Aug. 2019, pp. 35–38. doi: 10.1145/3340435.3342718.

[36] S. S. Gill et al., "Transformative effects of ChatGPT on modern education: Emerging Era of AI Chatbots," Internet of Things and Cyber-Physical Systems, vol. 4, pp. 19–23, Jan. 2024, doi: 10.1016/j.iotcps.2023.06.002.

[37] R. Tiwari, "The Impact of AI and Machine Learning on Job Displacement and Employment Opportunities," INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT, vol. 07, no. 01, Jan. 2023, doi: 10.55041/ijsrem17506.

[38] Yaou Hu a and Hyounae (Kelly) Min b, "The dark side of artificial intelligence in service: The 'watching-eye' effect and privacy concerns," Int J Hosp Manag, vol. 110, Apr. 2023.

[39] P. Walton, "The limitations of decision-making," Information (Switzerland), vol. 11, no. 12, pp. 1–22, Dec. 2020, doi: 10.3390/info11120559.

[40] J. Finnie-Ansley, P. Denny, A. Luxton-Reilly, E. A. Santos, J. Prather, and B. A. Becker, "My AI Wants to Know if This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises," in ACM International Conference Proceeding Series, Association for Computing Machinery, Jan. 2023, pp. 97–104. doi: 10.1145/3576123.3576134.

[41] A. R. Deekshitha Arasa Shriyanka Jamadade, "A Survey on Artificial Intelligence (Window to Mankind)." [Online]. Available: www.ijert.org