INFLUENCE OF ARTIFICIAL INTELLIGENCE ON THE EVOLUTION OF CODING LANGUAGES: A REVIEW

Ankit Singh[1], Dipti Ranjan Tiwari[2]

[1]Master of Technology, Computer Science and Engineering, Lucknow Institute of Technology, Lucknow, India
[2]Assistant Professor, Department of Computer Science and Engineering, Lucknow Institute of Technology, Lucknow, India

---***---

Abstract - *This comprehensive review paper thoroughly examines the intricate relationship between artificial intelligence (AI) and the evolution of coding languages. Through a detailed analysis of various literature and case studies, it uncovers the significant impact of AI technologies on the development of programming languages. The paper traces the historical origins of AI in coding languages and delves into the latest advancements that have reshaped the landscape of software development. It highlights how AI-driven tools and methods have revolutionized programming approaches, from compilers to interpreters, and programming environments. The incorporation of AI algorithms in these aspects has shown immense potential in optimizing code performance and enhancing developer efficiency. By exploring these advancements, the paper sheds light on the transformative power of AI in the coding world. Additionally, it addresses the complexities and ethical dilemmas that accompany the evolution of AI-driven languages, providing valuable insights for future research in this rapidly evolving field. The review paper serves as a comprehensive guide to understanding the impact of AI on coding languages and the opportunities it presents for innovation in software development.*

Key Words: Artificial intelligence (AI), Coding languages, Evolution, Software development, Machine learning, Programming paradigms.

1.HISTORY

The historical narrative of the convergence of artificial intelligence (AI) and programming languages is a saga of ingenuity, cooperation, and progression. It commenced in the 1950s when visionaries such as John McCarthy and Marvin Minsky laid the foundation for AI using languages like LISP. This marked the inception of AI programming, where languages were meticulously designed for AI operations. As AI research advanced, novel coding languages surfaced, such as Prolog, customized for logic programming. In the 1980s and 1990s, with the advent of expert systems, languages like C and C++ gained eminence for their efficacy in implementing AI algorithms. The 21st century witnessed a resurgence of enthusiasm for AI, propelled by advancements in machine learning and deep learning. Python, renowned for its user-friendly nature and robust libraries like TensorFlow and PyTorch, emerged as the predominant language for AI development. Presently, AI and programming languages continue to intersect, with ongoing endeavors to develop more specialized languages optimized for distinct AI operations, as well as incorporating AI functionalities into mainstream languages to democratize AI development. This extensive chronicle underscores the mutually beneficial relationship between AI and programming languages, molding the technological landscape and ushering in a realm of future innovations.

2.AI's Impact On Coding Language Evolution

Artificial intelligence (AI) has had a profound impact on the development of coding languages, fundamentally altering the approach that developers take to problem-solving and algorithm creation. One notable consequence is the growing need for languages specifically tailored for machine learning and data analysis tasks. Python, renowned for its user-friendly nature and extensive library support such as TensorFlow and PyTorch, has emerged as the predominant language in AI development. The proliferation of AI has also catalyzed innovations in language architecture, resulting in the emergence of novel languages and frameworks designed to tackle unique AI-related challenges. For example, languages like Julia have garnered attention for their exceptional performance in scientific computing, while languages like R have become indispensable for statistical analysis and data visualization. Furthermore, AI-powered tools like code completion and refactoring assistants, fueled by machine learning algorithms, are revolutionizing the coding process by streamlining development efforts and increasing accessibility. As AI technology progresses, coding languages are likely to undergo further evolution, incorporating more advanced AI capabilities to automate tasks, boost efficiency, and unlock novel opportunities in software development.

2.1.Importance of understanding AI's impact on coding language evolution

Comprehending the influence of AI on the evolution of coding languages is essential for developers and technologists alike, given its profound impact on software development practices. Initially, it offers insights into the evolving landscape of programming languages, allowing developers to keep pace with emerging trends and technologies. By understanding how AI influences language design and usage patterns, developers can make well-informed decisions regarding language selection, tooling adoption, and skill development to remain competitive in the fast-changing tech industry. Additionally, grasping the impact of AI on coding languages stimulates innovation by sparking the creation of new languages, frameworks, and tools designed to tackle emerging AI challenges and opportunities. This knowledge equips developers to effectively utilize AI-driven solutions, enhancing the performance, scalability, and maintainability of their codebases. Moreover, it promotes interdisciplinary collaboration between AI researchers and programming language designers, encouraging the development of hybrid approaches that combine the strengths of AI and traditional programming paradigms. Ultimately, a profound understanding of AI's influence on the evolution of coding languages empowers developers to fully leverage the potential of AI technologies, propelling advancements in software development and shaping the future of technology.

3.TRADITIONAL CODING LANGUAGES

Traditional coding languages are the established programming languages that have been utilized for many years before the rise of artificial intelligence (AI) in the tech industry.

These languages, such as C, C++, Java, JavaScript, Python, and Ruby, were created to cater to different application domains like system programming, web development, and enterprise software.

They are known for their syntax, semantics, and programming paradigms, which can be procedural, object-oriented, functional, or declarative.

It is crucial to understand how AI has impacted traditional coding languages, as AI technologies have influenced the way these languages are used and expanded.

For instance, integrating AI libraries and frameworks into languages like Python has allowed developers to incorporate machine learning, natural language processing, and computer vision capabilities seamlessly into their existing codebases.

This integration has opened up new possibilities for developers to enhance their projects and build more advanced applications.

Top Programming Languages Devs Should Be Looking At
PHP
Swift
C#
Java
JS
Python
Kotlin
R
C/C++
GO

Figure-1: Coding Languages

The integration of AI-driven tools, such as static code analysis, automated testing, and code generation, has significantly improved the productivity and efficiency of developers who work with conventional programming languages. With the continuous advancement of AI technology, traditional coding languages are also evolving to integrate AI-specific functionalities and enhancements. For example, newer versions of programming languages are now equipped with features like built-in support for parallel computing, distributed systems, and optimized algorithms, which are crucial for the development of large-scale AI applications. It is imperative for developers to comprehend the influence of AI on traditional coding languages in order to effectively utilize AI tools, enhance their programming abilities, and stay abreast of the changing landscape of the software development industry.

4.EMERGENCE OF ARTIFICIAL INTELLIGENCE AND ITS INITIAL IMPACT ON CODING LANGUAGES

The emergence of artificial intelligence (AI) has significantly impacted coding languages in various ways:

4.1.Specialized AI Languages

In recent years, there has been a noticeable increase in the creation of specialized programming languages that are specifically designed for artificial intelligence (AI) and machine learning purposes. These languages, such as Python, R, and Julia, have become increasingly popular because of their vast libraries and frameworks that are customized for AI development. This trend highlights the growing importance and demand for languages that cater to the unique needs of AI and machine learning projects.

4.2.Integration of AI Features

In recent years, mainstream programming languages such as Python, Java, and C++ have begun incorporating AI-related features and libraries to enhance their capabilities. For instance, Python boasts popular libraries like TensorFlow and PyTorch that cater to deep learning tasks, scikit-learn for machine learning purposes, and NLTK for natural language processing applications. This integration of AI functionalities into these languages has opened up new possibilities for developers to create sophisticated and intelligent software solutions. The availability of these libraries within the ecosystems of Python, Java, and C++ has made it easier for programmers to implement AI algorithms and techniques seamlessly, thereby accelerating the development of AI-powered applications across various domains. As a result, the intersection of AI and mainstream programming languages has significantly contributed to the advancement of technology and innovation in the digital era.

4.3.Automated Code Generation

Artificial intelligence (AI) has revolutionized the way software development is done by introducing tools that have the ability to automatically generate code based on high-level descriptions or requirements provided by developers. These tools leverage advanced techniques like natural language processing (NLP) and machine learning to comprehend the specifications given to them, enabling them to not only generate code snippets but also entire programs. This advancement in technology has greatly improved the efficiency and speed of software development, allowing developers to focus more on the creative aspects of their work rather than the repetitive and time-consuming task of coding. With AI-powered code generation tools, developers can now bring their ideas to life more quickly and accurately than ever before.

4.4.Enhanced Development Tools

Artificial intelligence (AI) has revolutionized the way programmers write code by providing powerful development tools that can significantly enhance productivity and efficiency. One of the key advancements in this field is the integration of AI algorithms into integrated development environments (IDEs) to offer features like code suggestion and auto-completion. These AI-powered tools analyze coding patterns, understand context, and provide programmers with relevant suggestions to speed up the coding process and ensure the quality of the code being written. By leveraging the capabilities of AI, developers can now write better code in less time, leading to improved software quality and faster development cycles.

4.5.Optimization and Performance

Artificial intelligence (AI) techniques are currently being utilized in the field of software development to enhance code optimization and improve overall performance. These techniques encompass a range of functionalities such as automatic parallelization, memory management, and algorithm optimization. By implementing AI-driven solutions, developers can significantly boost the efficiency of their programs without the need for manual intervention. This integration of AI into code optimization processes marks a significant advancement in the realm of software engineering, offering developers innovative tools to streamline their workflows and deliver high-quality, high-performance software products to end-users.

4.6.Natural Language Programming

Artificial intelligence (AI) has revolutionized the field of natural language programming by allowing developers to write code using human language. This breakthrough simplifies complex programming tasks, making it more accessible to individuals who may not have formal training in coding. With AI, the barrier to entry into the world of programming has been significantly lowered, opening up new opportunities for innovation and creativity. This technological advancement has the potential to empower a wider range of individuals to engage in software development and contribute to the ever-evolving digital landscape.

5.APPLICATIONS OF AI IN CODING LANGUAGES

Artificial intelligence (AI) has a multitude of applications in programming languages, elevating different facets of software development. Presented below are some noteworthy examples:

5.1.Code Generation

Artificial Intelligence (AI) has the capability to automatically create code by interpreting high-level requirements or specifications. This process can involve generating anything from basic code snippets to complete programs. AI models, including language models and neural networks, have the ability to analyze existing codebases and then produce code that is both syntactically accurate and semantically meaningful. By leveraging AI in this way, developers can streamline the coding process and potentially increase efficiency in software development projects.

5.2.Code Optimization

Artificial intelligence (AI) techniques have the ability to enhance code in terms of performance, efficiency, and resource utilization. Through the use of AI algorithms, code patterns can be thoroughly analyzed to pinpoint bottlenecks and recommend various optimizations. These optimizations may include enhancements to algorithms, implementation of parallelization, or utilization of memory management techniques. By leveraging AI in the optimization process, developers can significantly improve the overall quality and effectiveness of their code, leading to better software performance and resource utilization.

5.3.Bug Detection and Debugging

Artificial intelligence (AI) powered tools have the capability to aid in the detection and resolution of software bugs. These tools work by analyzing various aspects of the code such as structures, execution traces, and runtime behavior. Through the use of sophisticated AI algorithms, potential issues can be pinpointed, fixes can be recommended, and in some cases, bugs can even be automatically patched within the codebase. This advanced technology is revolutionizing the way developers approach bug detection and debugging processes, ultimately leading to more efficient and reliable software development practices.

5.4.Code Refactoring

Artificial intelligence (AI) has the capability to assist developers in the process of refactoring code by automatically reorganizing and enhancing its structure without altering its outward functionality. This involves a range of tasks like adjusting variable names, isolating methods, streamlining imports, and simplifying intricate expressions. By leveraging AI technology, developers can efficiently enhance the overall quality and readability of their codebase, leading to more maintainable and scalable software solutions.

5.5.Code Completion and Suggestions

Artificial intelligence-powered code editors and integrated development environments (IDEs) are equipped with advanced capabilities that go beyond simple text editing. By harnessing the power of AI, these tools offer intelligent code completion and suggestion features that significantly enhance the coding experience for developers. One of the key benefits of AI-driven code editors is their ability to analyze the context of the code being written. This means that the tools can understand the structure and purpose of the code, allowing them to suggest relevant keywords, functions, or even entire code snippets. By doing so, developers can save time and effort, as well as reduce the likelihood of errors in their code. In addition to providing suggestions, AI-driven code editors can also offer real-time feedback on code quality and best practices. This feedback can help developers improve their coding skills by pointing out potential issues or areas for optimization. Furthermore, these tools can adapt to the individual coding style of each developer, providing personalized recommendations that cater to their specific needs. AI-driven code editors and IDEs represent a significant advancement in the field of software development. By leveraging the capabilities of artificial intelligence, these tools have the potential to revolutionize the way code is written, making the process more efficient, accurate, and enjoyable for developers.

5.6.Natural Language Programming

Artificial intelligence (AI) has revolutionized the way developers interact with code through the advent of natural language programming. This innovative approach allows developers to communicate with computer systems using human language, making the coding process more accessible and intuitive. Natural language processing (NLP) techniques play a crucial role in this process, as they are used to analyze and interpret natural language commands or queries. These commands are then translated into executable code, enabling developers to seamlessly transform their ideas and instructions into functional software applications. Overall, AI-powered natural language programming has significantly enhanced the efficiency and user-friendliness of the coding process, opening up new possibilities for collaboration and innovation in the field of software development.

5.7.Code Summarization and Documentation

Artificial intelligence (AI) models have the ability to automatically create summaries and documentation for codebases. Through the analysis of code structure, comments, and usage patterns, AI algorithms are capable of generating concise summaries, detailed function descriptions, or comprehensive documentation pages. This technology assists developers in comprehending and navigating code more effectively and efficiently.

5.8.Automated Testing and Quality Assurance

Artificial Intelligence (AI) has the capability to revolutionize software testing procedures through the automation of various tasks. By utilizing AI algorithms, test cases can be automatically generated, edge cases can be identified, and potential failures can be predicted with higher accuracy. AI-driven testing tools have the ability to enhance test coverage by exploring different scenarios, detect regressions by comparing current and previous test results, and ultimately ensure the reliability and robustness of software systems. With AI technology, the testing process becomes more efficient, thorough, and effective, ultimately leading to higher quality software products.

6.LITERATURE SURVEY

In this segment of the literature review, we have examined the prior scholarly endeavors surrounding the fusion of artificial intelligence with programming languages, and a synopsis of the aforementioned research is presented as follows:

Praveendra. This paper delves into the intriguing pattern that has been uncovered through research that delves into the intersection of artificial intelligence and sound symbolism. It has been found that machine learning algorithms are highly adept at learning sound symbolism, however, they have a tendency to exhibit a bias towards one category over the other. This phenomenon sheds light on the complex relationship between AI and language, highlighting the need for further exploration and understanding in this field.

Ermira et al. Artificial intelligence, such as the highly advanced GPT-3 developed by OpenAI, plays a significant role in shaping the evolution of coding languages. By improving language understanding and fostering creativity, AI technologies like GPT-3 are revolutionizing the way we interact with machines and raising intriguing questions about the future of human-machine symbiosis. The emergence of various creative and business applications powered by natural language processing (NLP) engines, exemplified by OpenAI's GPT-3, is sparking a new wave of discussions on the intricate relationship between humans and machines. As we hurtle towards the future, the dynamics of communication between humans and AI are constantly evolving, paving the way for a future where the boundaries between man and machine become increasingly blurred.

Jess et al. The authors of this paper delve into the social implications of a widely used AI tool known as algorithmic response suggestions, also referred to as "smart replies". Through their research, they discovered that the utilization of algorithmic responses leads to enhanced communication efficiency, a higher frequency of positive emotional language being used, and more favorable evaluations from communication partners. This sheds light on how AI technology can impact interpersonal interactions in a positive manner by streamlining communication processes and fostering a more positive tone in conversations.

Adetiba et al. The field of AI programming languages, including popular ones like LISP, PROLOG, and others, has undergone significant evolution over the years. These advancements have been strongly influenced by the progress made in Artificial Intelligence, which has played a key role in shaping the capabilities, limitations, and applications of these languages across different domains. This paper presents a thorough systematic literature review that delves into the evolution of AI programming languages, offering valuable insights into the year of implementation, development teams involved, as well as the specific capabilities, limitations, and applications associated with each language discussed. This comprehensive analysis sheds light on the progression of AI programming languages and their impact on the broader field of Artificial Intelligence.

Sayat et al. This research delves into the geographical aspect of language diffusion within a confined space, analyzing how different language species spread at a small scale. The study demonstrates that the proposed framework effectively mirrors real-world linguistic patterns, which are influenced by both the Turing instability and a process of spontaneous pattern formation seen in various natural systems. This work sheds light on the intricate dynamics of language evolution and distribution, offering insights into how languages evolve and adapt within specific environments.

Urma. Artificial intelligence plays a significant role in shaping the evolution of coding languages by driving changes in response to various factors such as user needs, advancements in hardware, and developments in research. This dynamic process often necessitates developers to constantly adapt their code in order to support the new versions of programming languages. In addition to this, research suggests that developers can benefit from machine support that specifically focuses on the search aspect of how programming languages evolve over time. As a practical example of this concept, a new optional run-time type system has been designed for Python. This innovative system allows developers to manually specify contracts, enabling them to identify and address semantic incompatibilities between Python 2 and Python 3. By incorporating machine support and implementing new tools like the run-time type system, developers can navigate the complexities of language evolution more effectively and efficiently.

Risto & Xun. The experimental findings indicate that the Evolutionary Reinforcement Learning-based Population Observation Model (ERL-POM) has proven to be successful in simulating the evolution of language in a specific context. This research demonstrates that artificial languages can develop and evolve within a controlled environment, particularly when communication is a crucial component in completing tasks assigned to the agents. The study highlights the significance of communication in driving the evolution of language and emphasizes the importance of situational context in language development.

Silvia. This paper delves into the fascinating realm of applying Darwinian principles to the evolution of programming languages. By exploring the extent to which biological evolutionary mechanisms can be adapted to the world of computer languages, it reveals that there are indeed several key evolutionary building blocks that can be identified in this field. Through a detailed analysis and comparison, it becomes evident that the evolution of programming languages follows a similar pattern to that of biological evolution, showcasing the remarkable adaptability and innovation that occurs within the world of computer science.

Ruoxuan. The authors conducted a comprehensive analysis of AI language from various perspectives, including linguistics and philosophy. Through this analysis, they identified the shortcomings in AI language and highlighted areas that require improvement. Additionally, the authors provided valuable insights and recommendations on enhancing the intelligence levels of current AI language. Their suggestions aim to propel AI language to new heights of sophistication and effectiveness in today's technological landscape.

Samuel. As software continues to advance in complexity, the integration of artificial intelligence has become crucial in streamlining and automating coding processes, as highlighted by the authors. This innovative technology is proving to be a valuable tool in simplifying the intricacies of coding tasks, ultimately enhancing the overall efficiency of software development. By harnessing the power of artificial intelligence, developers can now tackle complex coding challenges with greater ease and precision, leading to more robust and reliable software products. The implementation of AI-driven solutions is paving the way for a new era in software development, where productivity and quality are significantly improved through intelligent automation.

Chowdhary. This paper introduces a series of innovative experimental languages that have the potential to dominate the programming language landscape in the era of new multi-core architectures. The aim is to provide compelling evidence that these new languages are not only more resilient than their predecessors but also successfully incorporate a blend of features from older languages. It is believed that these languages will revolutionize the way software is developed and pave the way for more efficient and effective programming practices in the future.

Zejie et al. The Abrams-Strogatz model of language competition, originally designed for two languages, has been expanded to consider multiple languages in a new study. This model was put to the test by examining the shifts in language usage behaviors over the past few decades in both Singapore and Hong Kong. The research revealed the presence of tipping points in language dominance, where multiple attractors were identified. This suggests that the dynamics of language competition in these regions are complex and influenced by various factors. The findings shed light on the intricate nature of language evolution and the importance of considering multiple languages in studying language dynamics.

Erik et al. This research project utilizes both manual techniques and non-GP Artificial Intelligence approaches to extract valuable insights from synthesis problem definitions. These insights are then leveraged to inform the development of the grammar employed by Grammatical Evolution, as well as to enhance its fitness function. By combining various methods and technologies, this work aims to optimize the performance and effectiveness of the Grammatical Evolution system through the integration of knowledge extracted from problem definitions.

Huang et.al: In this research paper, a novel method is introduced for designing error correction codes (ECC) using artificial intelligence technology. The approach involves utilizing AI algorithms to construct the error correction codes, while also incorporating a code evaluator to measure the performance metrics of the codes. This innovative technique aims to enhance the efficiency and accuracy of error correction in communication systems by harnessing the power of artificial intelligence. By leveraging AI-driven solutions, this research opens up new possibilities for improving the reliability and effectiveness of error correction mechanisms in various applications.

Luc. This report provides an in-depth analysis of a significant and ongoing project that focuses on conducting extensive experiments with evolutionary language games. The project aims to explore the dynamics of language evolution over an extended period of time and has already yielded substantial findings. The report delves into the key outcomes and milestones that have been reached thus far, shedding light on the progress and impact of the research endeavor.

Rob. The computational model of an artificial creative system discussed in this study builds upon existing models by incorporating a linguistic component. This linguistic component plays a crucial role in facilitating the creation and dissemination of artistic works along with their accompanying descriptions. By integrating language into the computational framework, the system is able to generate a richer and more nuanced understanding of the creative process. This innovative approach not only enhances the system's ability to produce original works but also enables it to effectively communicate the intent and context behind each piece of art. Ultimately, the inclusion of a linguistic component in the computational model of artificial creative systems opens up new possibilities for exploring the intersection of language and creativity in the digital age.

Peter & Rowlands. The theory of nilpotent quantum mechanics, which is discussed in the following paragraphs, has the potential to provide physicists with the long-awaited 'Theory of Everything'. This theory is based on the NUCRS foundational universal organization principle, which aims to unify all fundamental forces and particles in the universe. By incorporating the concept of nilpotency into quantum mechanics, scientists hope to develop a comprehensive framework that can explain the behavior of matter and energy on both the microscopic and macroscopic scales. This groundbreaking approach could revolutionize our understanding of the physical world and lead to significant advancements in various scientific fields.

Ana Maria. This research paper delved into the intricate relationship between artificial intelligence (AI) and language acquisition within the framework of the education 4.0 era. It emphasized the transformative impact of AI on language learning by introducing cutting-edge technologies and innovative approaches in the field of education. The paper thoroughly analyzed how AI is reshaping the landscape of language education, paving the way for more personalized and effective learning experiences for students. Furthermore, it discussed the implications of integrating AI into language learning curricula, exploring the potential benefits and challenges that arise from this technological advancement. Overall, the paper provided a comprehensive overview of the intersection between AI and language education in the context of the evolving educational landscape.

7.CONCLUSION

In conclusion, the symbiotic relationship between artificial intelligence (AI) and coding languages has significantly transformed the programming landscape. Our thorough examination in this comprehensive review paper demonstrates that AI technologies have not only changed the way code is written, but have also introduced entirely new programming paradigms. The impact of AI on coding languages is profound, with the development of AI-powered tools for code generation and optimization, as well as the creation of languages tailored for machine learning and data science. The integration of AI and coding languages is expected to deepen further as AI capabilities advance, resulting in the establishment of more intelligent, adaptive, and human-like programming environments. Nevertheless, this evolution comes with challenges, such as ethical considerations related to AI-generated code and the need for developers to continuously learn and adapt. Successfully navigating these challenges while seizing the opportunities offered by AI is essential for driving the ongoing evolution of coding languages towards a future that is characterized by efficiency, inclusivity, and innovation.

REFERENCE

1. Jess, Hohenstein., Dominic, DiFranzo., René, F., Kizilcec., Zhila, Aghajari., Hannah, Mieczkowski., Karen, Levy., Mor, Naaman., Jeffrey, T., Hancock., Malte, F., Jung. (2021). Artificial intelligence in communication impacts language and social relationships.. arXiv: Human-Computer Interaction,

2. Emmanuel, Adetiba., Temitope, M., John., Adekunle, Akinrinmade., Funmilayo, S., Moninuola., Oladipupo, Akintade., Joke, A., Badejo. (2021). Evolution of artificial intelligence languages, a systematic literature review.. arXiv: Artificial Intelligence,

3. Kornack and P. Rakic, "Cell Proliferation without Neurogenesis in Adult Primate Neocortex," Science, vol. 294, Dec. 2001, pp. 2127-2130, doi:10.1126/science.1065467.

4. M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

5. R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

6. Coulin, C., Zowghi, D., & Sahraoui, A. (2010). MUSTER: A Situational Tool for Requirements Elicitation. In F. Meziane, & S. Vadera (Eds.), Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects (pp. 146-165)

7. Harmain, H. M., & Gaizauskas, R. (2003). CM-Builder: A natural language-based CASE tool for object-oriented analysis. Automated Software Engineering Journal, 10(2), 157–181

8. Hewett, Micheal, and Rattikorn Hewett (1994). 1994 IEEE 10th Conference on Artificial Intelligence for Applications.

9. Hull, E., Jackson, K., & Dick, J. (2005). Requirements Engineering. Berlin: Springer.

10. Kof, L. (2010). From Textual Scenarios to Message Sequence Charts. In F. Meziane, & S. Vadera (Eds.), Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects (pp. 83-105).

11. Smith, T. J. (1993). READS: a requirements engineering tool. Proceedings of IEEE International Symposium on Requirements Engineering, (pp. 94–97), San Diego. SSBSE (2010). http://www.ssbse.org, checked 10.5.2011.

12. Vadera, S., & Meziane, F. (1994). From English to Formal Specifications. The Computer Journal, 37(9), 753–763.

13. George F Ludger "Artificial Intelligence Structure and strategies for complex problem solving" 5th Edition Pearson,2009

14. Xindong Wu, Senior Member, IEEE" Data Mining: An AI Perspective"1965 [4]Holland, "Adaption in Nature and Artificial Systen"1965.

15. Searle (1990).The Brain Mind Computer Program? Scientific America, 262,pp.

16. T kamba "A Web Marketing With Automatic Pricing" Computer network vol 33 775-788(2000)

17. Kevin Warwick "Artificial Intelligence: The Basic"2011.

18. John E.Kelly, "Smart Machine-IBM'S Watson And The Era of Cognitive Computing" 2013.